Departamento de Ingeniería de Sistemas y Automática

PROYECTO FIN DE CARRERA

DISEÑO HARDWARE Y SOFTWARE DE UNA FUENTE ORNAMENTAL

Autor: Berta Gutiérrez Montes
Tutor: José María Armingol Moreno

Leganés, Diciembre de 2012

Título: DISEÑO HARDWARE Y SOFTWARE DE UNA FUENTE ORNAMENTAL
Autor: Berta Gutiérrez Montes
Director: José María Armingol Moreno

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis padres, por haberme apoyado siempre en cada paso que he dado, por la confianza que tenéis en mi y en las decisiones que tomo.

A mi hermano y a esa familia que he tenido el privilegio de elegir: mis amigos.

A Pablo.

Agradezco a Roberto Sanz la grandísima oportunidad que me dió. Y por supuesto este proyecto no hubiera sido lo mismo sin Félix Marín. Gracias Féllix.

A José María Armingol, por ayudarme a encontrar el final de esta carrera que a veces parecía que no iba a llegar nunca.

Por último mi más sincero agradecimiento a José Manuel Pérez García, quién fue mi tutor, al que recurrí por lo buen profesor y buena persona que era.

Resumen

Este proyecto surge de la necesidad de modernizar el mobiliario urbano ya que sin un concepto moderno de automatización, la eficiencia energética no es posible.

El siguiente documento presenta el desarrollo tanto hardware cómo software de una fuente ornamental. Además de la realización del programa de funcionamiento de la fuente, se ha desarrollado un interfaz muy sencillo para la puesta en marcha, el control y el mantenimiento de la misma.

Los dos grandes retos de este proyecto eran conseguir una gran flexibilidad, abarcando un gran abanico de tipos de fuentes, en las que podría instalarse esta arquitectura, y que el manejo de la misma fuera muy intuitivo y asequible para cualquier persona.

Palabras clave: Variador de velocidad, comunicaciones, HMI, SoMachine, VijeoDesigner, Pantallas de operador.

Abstract

This master tesis arises from the need to modernize the street furniture because without a modern concept of automation, energy efficiency is not possible.

The following technical report presents the development of both the hardware and the software of an ornamental fountain. In addition to the implementation of the fountain operation program, we have developed a simple interface for setting up, monitoring and the maintenance of it.

The two great challenges of this project were to achieve greater flexibility, covering a wide range of types of fountains, in which could be installed this architecture, and the management of it was very intuitive and affordable for anyone.

Keywords: Variable speed drive, communications, HMI, SoMachine, VijeoDesigner, operator screens.

Índice general

Índice de figuras

Índice de tablas

Glosario

PLC:*Controlador lógico programable* (en inglés *Programmable Logic Controller*). Diseñados para programar y controlar procesos secuenciales en tiempo real. Son dipositivos electrónicos que reproducen programas informáticos.
HMI:*Human Machine Interface*. Interfaz de usuario, se usa para referirse a la interacción entre humanos y máquinas; Aplicable a sistemas de Automatización de procesos.
CANopen:protocolo de alto nivel basado en capas físicas CAN y desarrollado como una red de trabajo estandarizada con capacidades de configuración altamente flexibles.Originalmente diseñado para aplicaciones de control de movimiento.
SoMachine: Software de Schneider Electric.
OEM: Original Equipment Manufacturers, fabricantes de maquinaria.
Vijeo Designer: Software de Schneider Electric.
Modbus:protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs).
PC: Peronal Computer, ordenador personal. Es una microcomputadora diseñada en principio para ser usada por una sola persona a la vez.
CPU:*Central Processing Unit* (CPU/Unidad Central de Procesamiento) o simplemente el procesador o microprocesador, es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.
EPROM:*Erasable Programmable Read-Only Memory* (ROM programable borrable). Es un tipo de chip de memoria ROMno volátil.
EEPROM: (o E²PROM)*Electrically Erasable Programmable Read-Only Memory* (ROM programable y borrada eléctricamente). Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, a diferencia de la EPROM que ha de borrarse mediante un aparato que emite rayos ultravioleta. Son memorias no volátiles
IEC:*International Electrotechnical Commission* .La Comisión Electrotécnica Internacionales una organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas. Numerosas normas se desarrollan conjuntamente con la ISO (normas ISO/IEC).
LD:El LADDER, también denominado lenguaje de contactos o en escalera, es un lenguaje de programación gráfico muy popular dentro de los autómatas programables debido a que está basado en los esquemas eléctricos de control clásicos

FBD:*Function Block Diagram.* Diagrama con bloques de función, lenguaje de programación basado en bloques que realizan operaciones matemáticas simples para poder determinar una salida
IL:*Instruction List.* La lista de instrucciones es un lenguaje de programación textual orientado a la máquina.
ST:*structured text.* El texto estructurado es unlenguaje de alto nivel estructurado por bloques que posee una sintaxis parecidaal PASCAL.
SFC:*Sequential function chart.*Es un leguaje de programación gráfico utilizado para **PLCs**. Es uno de los cinco lenguajes definidos en el estándarIEC 61131-3. Lenguaje basado en GRAFCET.
Grafcet:*GRAphe Functionel de Commande Etape Transition.*Es un grafo o diagrama funcional normalizado, que permite hacer un modelo del proceso a automatizar, contemplando entradas, acciones a realizar, y los procesos intermedios que provocan estas acciones.
SCADA:*Supervisory Control And Data Acquisition.*Supervisión, Control y Adquisición de Datos, es un software para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita retroalimentación en tiempo real con los dispositivos de campo, controlando el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo y permite su gestión e intervención
PAC: *Programmable Automation Controller*. Es un controlador de automatización programable, es una tecnología industrial orientada al control automatizado, al diseño de prototipos y a la medición. El PAC se refiere al conjunto formado por un controlador (una CPU típicamente), módulos de entradas y salidas, y uno o múltiples buses de datos que lo interconectan todo.
RTU: *Remote Terminal Unit.* Unidad Terminal Remota define a un dispositivo basado en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto donde se procese.
ASIC: *Circuito Integrado para Aplicaciones Específicas* es un circuito integrado hecho a la medida para un uso en particular, en vez de ser concebido para propósitos de uso general.
IPM: *Intelligent Power Module.* Modulos inteligentes de Potencia
LED: *Light-Emitting Diode. Es* un diodo semiconductor que emite luz.
IGBT: *Insulated Gate Bipolar Transistor. E*s un dispositivo semiconductor que generalmente se aplica como interruptor controlado en circuitos de electrónica de potencia.
OSI: *open system interconnection.* El modelo de interconexión de sistemas abiertos es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización (ISO) en el año 1984. Es decir, es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.
TCP: *Transmission Control Protocol*. Protocolo de comunicación orientado a conexión y fiable del nivel de transporte, actualmente documentado por IETF RFC 793. Es la capa intermedia entre el protocolo de internet (IP) y la aplicación.
ASCII: *American Standard Code for Information Interchange*. Código Estándar Estadounidense para el Intercambio de Información es un código de caracteres basado en el alfabeto latino.
CRC: *comprobación de redundancia cíclica* es un código de detección de errores usado frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos.
LRC: *Longitudinal Redundancy Checking* ("Chequeo de Redundancia Horizontal")

TCP/IP: se le denomina *conjunto de protocolos TCP/IP*, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).Ees un conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras.
CSMA/CR: *Carrier Sense Multiple Access with Collision Resolution*. Acceso múltiple con escucha de portadora con resolución de Colisiones, es un protocolo de acceso al medio compartido.
TDMA: *Time Division Multiple Access.* La multiplexación por división de tiempo es una técnica que permite la transmisión de señales digitales y cuya idea consiste en ocupar un canal (normalmente de gran capacidad) de transmisión a partir de distintas fuentes, de esta manera se logra un mejor aprovechamiento del medio de transmisión.
DeviceNet: es un protocolo de comunicación usado en la industria de la automatización para interconectar dispositivos de control para intercambio de datos
IP65: El Grado de proteccion IP hace referencia al estándar internacional IEC 60529 Degrees of Protection utilizado con mucha frecuencia en los datos técnicos de equipamiento eléctrico y/o electrónico (en general de uso industrial como sensores, medidores, controladores, etc).
RS485: Es un estándar de comunicaciones en bus de la capa física del Modelo OSI.
Ethernet: es un estándar de redes de área local para computadores con acceso al medio por contienda CSMA/CD. Ethernet define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.
Codesys: es un entorno de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3. El término CoDeSys es un acrónimo y significa Sistema de Desarrollo de Controladores.
POU: *Program Organization Unit.* Secciones de programa independientes.
GVL: *Global variable list.*
MAST: Tarea principal del proyecto.
DUT: *Data Unit Type. Tipo* de dato creado por el usuario.
FAST: Tarea rápida. Interrupciones.
XML*: eXtensible Markup Language*. Es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes.
E/S: Entradas y Salidas
PDO: Process Data Object

Capítulo 1

Introducción y objetivos

1.1 Introducción

El siguiente Proyecto Fin de Carrera tiene como objeto el diseño tanto hardware como software de una fuente ornamental.

Este proyecto surge como consecuencia de la descomercialización de un PLC de la marca Schneider Electric, que utilizaba siempre un fabricante de fuentes ornamentales. El proyecto abarca, además del diseño de una nueva arquitectura hardware que satisfaga todo lo que tenía la arquitectura anterior, el desarrollo de toda la programación. Esta combinación, da lugar a un interesante y complejo proyecto, dónde poder ver la aplicación directa de muchos de los conocimientos adquiridos a lo largo de la carrera.

El resultado de este proyecto no es sólo la migración de un sistema a otro. La configuración anterior era extremadamente sencilla en cuanto a funcionalidades, y aprovechando el desafío que supone un cambio de este tipo, era una buena oportunidad para adaptar sus fuentes a los tiempos que corren añadiendo algunos conceptos cómo son las pantallas HMI, para un mejor manejo y puesta en marcha para los operarios, o tratar de sacarle un mayor partido a la capacidad de los nuevos autómatas, mejorando por ejemplo los juegos de chorros de agua con las luces.

Su configuración era muy sencilla en cuanto a funcionalidades y sin embargo con un cableado muy complejo, también se ha tratado de simplificar el esquema eléctrico en la medida de lo posible.

A grandes rasgos, la nueva arquitectura estará gobernada por el autómata M238 de la plataforma SoMachine. Éste comunicará vía CANopen con hasta ocho variadores ATV312, y el operario podrá interactuar con el sistema a través de una pantalla táctil HMI855.

Tanto los variadores como la pantalla son dispositivos de la misma plataforma SoMachine, esto va a ofrecer muchas ventajas que ya se verán más adelante.

Esta plataforma ha sido diseñada con la intención de aglutinar la mayoría de los dispositivos que suelen necesitar los OEMs (fabricantes de maquinaria), simplificando mucho todo lo relativo a las comunicaciones entre dispositivos.

También pertenecen a la plataforma SoMachine los dos software que se han utilizado y que se estudiaran en detalle, el SoMachine 3.0 y el VijeoDesigner V6.0 para la pantalla.

En este documento se va a realizar primeramente una introducción al campo de la automatización e información sobre las materias necesarias para poder facilitar al lector la comprensión de la extensión del proyecto.

A partir de los requisitos de funcionamiento se eligen los elementos hardware y se detallará la programación realizada para que la máquina cumpla con las especificaciones técnicas que se requieren.

1.2 Objetivos

Este proyecto engloba la monitorización, control y automatización de un gran abanico de fuentes ornamentales.

El objetivo de este proyecto es el de realizar el diseño tanto hardware como software de una fuente ornamental utilizando las herramientas de Schneider Electric de la plataforma de automatización SoMachine.

Este objetivo puede desglosarse en tres objetivos más concretos:

- Detallar la arquitectura de la máquina, y presentar las diferentes tecnologías que engloba.
- Programar el software de control necesario para el correcto funcionamiento de la fuente en base a los requisitos del fabricante.
- Diseñar e implementar los sinópticos del panel HMI incorporado para el control y monitorización del funcionamiento.

Dentro de esta idea global surgen, por supuesto, otros objetivos complementarios:

1. Eficiencia energética de la fuente, requisito fundamental en la situación actual.

2. Robustez de la arquitectura. El sistema no puede fallar, y las condiciones del entorno de una fuente ornamental pueden ser adversas.

3. Fácil manejo para la puesta en marcha y el mantenimiento de la fuente. Desarrollo de un interfaz Hombre-Máquina sencillo y amigable.

4. Máxima flexibilidad posible. Se pretende realizar un programa que sea válido, sin modificaciones de código, para un gran abanico de fuentes.

1.3 Fases de desarrollo

En primer lugar fue necesario conocer a fondo la aplicación existente, y estudiar la posibilidad de adaptarla a la nueva gama de PLCs.

Los nuevos equipos además de poseer mayores capacidades y funcionalidades, son menos económicos. Para poder realizar una arquitectura competitiva era necesario eliminar algunos componentes de la arquitecturaanterior, por ejemplo se ha eliminado el reloj astronómico (que determina cada día del año en que momento exacto se hace de día y cuando de noche), con una sección de programa que, mediante un complejo algoritmo, realiza las funciones de este.

En esta etapa fueron necesarias varias reuniones con el fabricante de fuentes. También se estudió que ofrecían otros fabricantes, para aumentar las funcionalidades de las fuentes que estos realizaban, compensando de este modo la diferencia económica que pudiera haber.

Lo siguiente fue definir la arquitectura hardware inicial, que posteriormente se fue mejorando esquivando las limitaciones que aparecían:

Inicialmente, la arquitectura consistía en una pantalla con PLC integrado, la XBTGC concretamente, ésta comunicaba a través de Modbus con los variadores, y eran necesarios algunos módulos con entradas y salidas digitales adicionales. Tras realizar algunas pruebas se llega a la conclusión de que este bus de comunicaciones no aportaba la robustez necesaria ya que si caía alguno de los variadores, cosa bastante frecuente en este tipo de aplicaciones, caía todo el bus.

Esto condujo a la determinación de que el bus que mejor encajaba era CANopen, esta modificación, además de cambiar aspectos de la configuración, parametrización, y de la programación, impedía mantener la arquitectura hardware inicial. Para utilizar la pantalla como maestro CANopen, era necesario un módulo extra, que impedía la conexión de los módulos adicionales de entradas y salidas.

Después de éste y algunos otros cambios se llega a la arquitectura hardware actual que se verá más adelante.

Otra de las fases de desarrollo fue la programación por un lado del PLC y por otro de la pantalla, sin olvidar nunca la relación entre ellas, aunque gracias a la plataforma SoMachine, el linkado de variables resulta muy sencillo.

Una última etapa en cuánto al desarrollo de esta aplicación fue la de pruebas con el fabricante, escuchando todas sus propuestas de mejora, y tratando de llevarlas a cabo.

Para terminar, y dejar constancia de todo el trabajo realizado, está la etapa de documentación. En esta etapa, no sólo está la redacción de este documento, sino también la creación de un documento que sirva de guía para la instalación, puesta en marcha y el manejo de los operarios, y la recolección de toda la documentación técnica relacionada con los equipos y tecnologías utilizadas en dicho proyecto.

1.4 Medios empleados

El laboratorio en el que se ha trabajado, disponía de un panel de pruebas con el PLC necesario para cada fase del proyecto, los variadores, la pantalla HMI, fuentes de alimentación, todos los dispositivos de campo necesarios (el anemómetro, sensores, pilotos…), todo el cableado, y los buses de comunicaciones. En dicho panel, se podía probar toda la arquitectura que se estuviera estudiando en cada momento.

Por otro lado, toda la programación la he realizado desde un PC en el que tenía instalados los softwares necesarios para ello.

1.5 Estructura de la memoria

Introducción y objetivos

Descripción global del problema del que surge este proyecto y motivaciones por las que se debe llevar a cabo el proyecto. También se incluye un breve desglose de las fases de desarrollo del proyecto.

Estado del arte de la automatización

Introducción al mundo de la automática y sus aplicaciones, desde sus orígenes hasta la actualidad.

Base teórica

Descripción de los conceptos y protocolos necesarios para el desarrollo y la mejor comprensión del proyecto.

Arquitectura y programación del proyecto

Descripción detallada de la arquitectura hardware desarrollada para el proyecto.

Descripción detallada del desarrollo software tanto del PLC como de la pantalla HMI y las herramientas utilizadas para ello en el proyecto.

Conclusiones y mejoras

Se detallan las conclusiones obtenidas tras la realización del proyecto y se exponen una serie de mejoras para proponérselas al fabricante.

Presupuesto

En este capítulo se calculan los costes del diseño y la programación, de una fuente ornamental estándar.

Bibliografía

Se detallan los diferentes recursos bibliográficos y electrónicos utilizados para la realización de este proyecto.

Anexos

Se detallan otras informaciones relevantes a la hora de comprender este proyecto en su totalidad.

Capítulo 2

Estado del arte de la automatización

Autómata.

(Del lat. automăta, t. f. de -tus, y este del gr. αυτοματος, espontáneo).

1. m. Instrumento o aparato que encierra dentro de sí el mecanismo que le imprime determinados movimientos.

2. m. Maquina que imita la figura y los movimientos de un ser animado.

3. m. coloq. Persona estúpida o excesivamente débil, que se deja dirigir por otra.

2.1 Autómatas y robots, entre mito y realidad.

Con los autómatas (Del griego *automatos,* es decir espontáneo) el hombre ha tratado de reproducir artificialmente el movimiento de los seres animados. La historia de los autómatas se remonta a un tiempo muy lejano. Hefesto, el dios de la metalurgia que forjaba armas para los dioses del Olimpo, creó jóvenes muchachas hechas de oro e inventó trípodes semimovientes [1]. Pausanias describió los autómatas construidos por el

mítico Dédalo[2] (520 a.C.); y también Píndaro, Apolonio de Tiana y Dion Casio hablaron de estatuas animadas.

Figura 2.1-1 *Detalle del mecanismo de Anticitera- un ordenador astronómico construido por los griegos en el año 80 a.C. Fue hallado en 1902 próximo a los restos del naufragio de una nave romana en las aguas que rodean la isla de Anticitera. Se trata de uno de los más importantes descubrimientos arquelógicos de la historia. El mecanismo está compuesto por treinta ruedas dispuestas de manera compleja, está hecho en bronce y se presupone que servía para representar fenómenos astronómicos*

Fue en Alejandría, durante el reinado de Ptolomeo II Filadelfo, donde Ctesibios estudió la neumática (ciencia que se ocupa del comportamiento de los gases) y describió la aplicación de los pistones en las catapultas en sus *Comentarios*. Al cargar las catapultas, los pistones emitían explosiones, llamas y humo: Ctesibios había descubierto el principio del motor diesel, o lo que es lo mismo, que los aceites comprimidos producen una llamarada. También inventó otras máquinas hidráulicas como la clepsidra de agua, los pistones con contrapeso, las bombas anti incendio y los sistemas de elevación de agua.

Filón de Bizancio fue sucesor y discípulo de Ctesibios. En sus obras describió una serie de máquinas de guerra muy admiradas. Fue el padre de la teoría de las palancas y utilizó con verdadera maestría la neumática, combinándola con el uso de la presión atmosférica y la presión del vapor de agua. Describió muchos autómatas construidos según estos principios, como, por ejemplo, fuentes con animales que beben y animales que cantan, máquinas automáticas para el abastecimiento de agua e, incluso, un teatro completamente automático. Pero el genio absoluto irremediablemente asociado a Leonardo Davinci es Herón, que vivió en Alejandría en el siglo I d.C. y que fue un extraordinario precursor de tecnologías que no fueron plenamente desarrolladas hasta muchos siglos después. Autor de numerosos tratados, Herón afirmó con rotundidad la necesidad de una preparación completa, suma de teoría y práctica, exactamente igual que Leonardo, quien escribió: “La práctica siempre debe construirse sobre una buena teórica” *(Manuscrito G,f. 8r)* y “la ciencia es el capitán, y la práctica son los soldados” *(Manuscrito I,f.130*r).

La *Pneumática* se abría con una introducción teórica seguida de una descripción de numerosos dispositivos accionados por la presión del agua, del vapor y del aire comprimido. En esta obra, el estudioso alejandrino reveló su capacidad como inventor describiendo dispositivos, tales como la eolípila y la llamada fuente de Herón. La eolípila, o esfera de Eolo, muestra cómo la energía térmica puede ser transformada en energía mecánica aprovechando la presión derivada del calentamiento del agua dentro de una esfera metálica. Se trataba de la primera máquina de vapor que realmente funcionaba

y estaba basada en los mismos principios físicos de las turbinas de vapor modernas. Herón dejó también un tratado sobre la construcción de máquinas de guerra. En su obra *Sobre los autómatas* ilustró teatrillos automáticos dotados de movimiento autónomo, rectilíneo o circular, durante todo el espectáculo. Sin embargo, la obra maestra de Herón fue su tratado de Mecánica. En este texto, el científico sistematizó definitivamente el aspecto teórico y práctico de la mecánica, reconduciéndola a las cinco máquinas simples (palanca, cabestrante, polea, tornillo y cuña), cuyo funcionamiento depende del principio de la palanca. También han sobrevivido fragmentos de la contribución de Herón a los relojes de agua y de sus *Comentarios a los Elementos de Euclídes.*

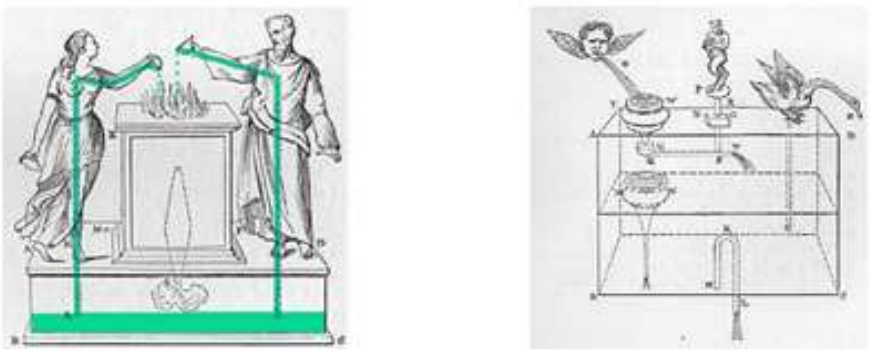


Figura 2.1-2 *a) Dos estatuas se mueven y vierten un líquido gracias a un sistema accionado por fuego y vapor. Herón, The Pneumatics.*

b) El autómata con forma de animal aspira el líquido que tiene delante gracias a un ingenioso sistema basado en la diferencia de presión a través de tubos, una doble cámara y un sifón. Herón, The Pneumatics.

Sus invenciones también fueron utilizadas en lugares de esparcimiento, como los jardines. Fueron célebres su *Fuente de la lechuza* y su estatua de *Artemisa Efesia.* La primera era una fuente cantora que, por medio del aire y el agua, reproducía el trino de los pájaros [3].

En más de una ocasión ha sido planteada la cuestión de por qué no se prosiguió con aquella "cultura de las máquinas" después de la cantidad de experiencias teóricas y prácticas que se habían conquistado en la época alejandrina, y de por qué para ello hubo que esperar hasta el siglo XVIII. El pensamiento científico griego ya había conquistado muchos de los principios teóricos, especialmente en mecánica, de los que la técnica moderna necesita: así que, si no existió una cultura real de las máquinas y sólo se trató de una utilización práctica destinada principalmente a producir juguetes ingeniosos (los considerados *thaumata,* es decir, milagros, objetos de maravilla), se debió todo a un conjunto de factores y condicionantes mentales, psicológicos, sociales y políticos. La máquina era admirada como un producto de ingenio, estaba destinada a fascinar y a maravillar, y no se veía la necesidad de que sustituyese el trabajo manual, ya que durante milenios la mano de obra servil estuvo disponible en gran cantidad y a buen precio. Hay que pensar también en la falta de medios de comunicación en aquella época y en que es un hecho que para que un invento se difunda y se aplique es necesario que sea reconocido y comprendido. Los descubrimientos van a la par que las necesidades de su propio tiempo, y aquello que se anticipa demasiado a su época, el fruto de mentes geniales y completamente fuera de lo común, no es comprendido por sus contemporáneos.

Del mundo griego pasamos al mundo árabe, siglos después. En el *Kitab al-Hiyal*, el libro de los mecanismos redactado por los hermanos Banu Musa en el siglo IX, aparecen ilustrados autómatas de distinto género. El matemático e ingeniero árabe Al-Jazari

describió varios autómatas, entre 1204 y 1206, en su *Libro de los conocimientos de los mecanismos ingeniosos.* En China, India, y Oriente Próximo se contaban historias fantásticas sobre autómatas.

Alberto Magno (siglo XIII), creador de un mítico sirviente automático, fue quien acuñó el término "androide", aquella máquina fue destruida por su célebre discípulo, Tomás de Aquino. Y parece ser que, ya en 1470, Johannes Muller, hizo volar un águila artificial para dar la bienvenida al emperador Maximiliano en su entrada a la ciudad de Nuremberg. [4]

2.2 La automatización del siglo XV en adelante.

La historia de los autómatas continuo con nombres como Jannello Torriani de Cremona (siglo XVI), que construyo diferentes autómatas para entretener al emperador Carlos V, o Salomón de Caus, que utilizó el agua como fuerza motora y reprodujo el sonido y movimiento de los pájaros. En 1680, el rey de Francia ordenó a Christian Huygens construir una máquina que representase todo un ejército de combate. Maillard ya en 1731, realizó mecanismos bastante sofisticados con engranajes y ruedas dentadas para reproducir le movimiento de los caballos. Y Jacques de Vaucanson realizó durante muchos años (hasta 1741) algunos de los más famosos autómatas de la historia y, de hecho, a él se le recuerda como el más célebre inventor de autómatas. Su *Pato artificial* en bronce dorado daba la impresión de beber, comer, aletear en el agua y digerir como un pato de verdad (el abdomen transparente permitía seguir el proceso desde el buche hasta el esfínter).

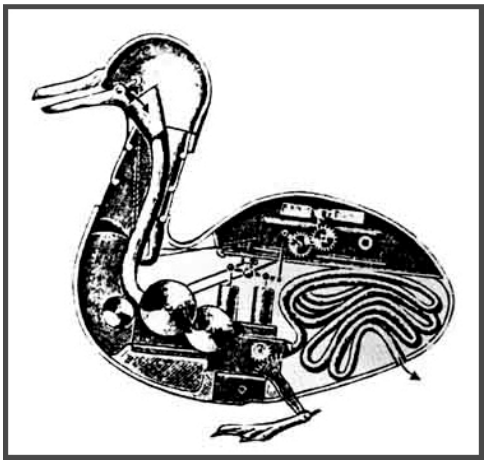


Figura 2.2-1 *El Pato artificial que realizó Vaucanson en 1738 bebía, comía y digería*

Su *Flautista* era un androide de 78 cm de alto que estaba sentado en una roca sobre un pedestal, como si fuese una estatua. El cofre que encerraba gran parte del mecanismo del motor contenía un cilindro de madera de 56 cm de diámetro y 83 cm de largo que giraba sobre su propio eje. Provisto de pernos, transmitía impulsos a quince palancas que por medio de cadenas e hilos, controlaban el nivel de aire en la recámara y el movimiento

de los labios, de la lengua y de las articulaciones de los dedos, que tocaban realmente la flauta con una variedad de doce melodías.

La creación del *Flautista* también sirvió para el estudio de la reparación humana. Nueve fuelles transmitían un chorro de aire más o menos débil a tres tubitos unidos a tres pequeñas recámaras situadas en el pecho del flautista. Allí los tubitos se unían hasta formar un único tubo que terminaba en la boca del flautista, cuyos labios dejaban pasar el aire en mayor o menor cantidad, según su apertura. Dentro de la cavidad bucal había una lengüeta móvil que se abría y se cerraba al pasar el aire.

También en el siglo XVIII, Pierre Jaquet-Droz construyo tres autómatas: el *Escritor*, el *Dibujante* y la *Pianista*, conservados hoy en el museo de Neuchâtel, en Suiza. El *Escritor* puede ser programado para escribir hasta cuarenta letras mojando la pluma en el tintero, el *Dibujante* realiza cuatro dibujos, y la *Pianista* toca un órgano pulsando las teclas con los dedos.

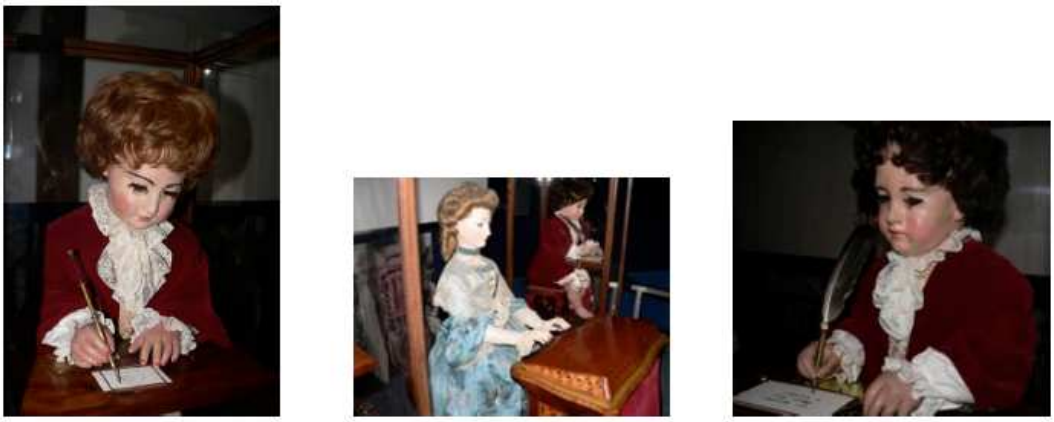


Figura 2.2-2 *"El dibujante", "La pianista" y "El escritor" de Jaquet-Droz.*

En el siglo XIX, Joseph Faber necesitó veinticinco años para realizar *Euphonia*, un rostro artificial que reproducía la voz humana, hacía preguntas y respondía, cantaba y reía, y hablaba en inglés con acento alemán. En 1893, George Moore construyó un androide que se movía gracias al vapor de un calentador de gas y que podía caminar a una velocidad de 15 km/h. A lo largo de ese siglo se realizaron muchos otros robots, fue su época dorada.

Pero los primeros autómatas sofisticados de verdad fueron creados durante la Ilustración gracias al arte de la relojería. En esta época en la que el saber científico y, sobre todo, la concepción biomecánica del ser humano predominaban fueron realizadas numerosas criaturas artificiales en un intento por imitar a la naturaleza lo más fielmente posible. Relojeros y mecánicos, cautivados por la medicina y las ciencias naturales, realizaron androides y animales automáticos para entretener y maravillar al público y para colaborar con el progreso de la ciencia. Todos los seres artificiales, por lo general, eran piezas únicas cuya realización requería un trabajo prolongado y difícil con resultados muy efectistas. En la primera mitad del siglo XIX, muchos de estos creadores de autómatas eran magos o se sentían atraídos por el ilusionismo, que por aquel entonces estaba muy de moda. Más tarde, con la revolución industrial se desarrollo una verdadera manufactura de autómatas y muñecos. Y finalmente, la Primera Guerra Mundial asestó el golpe definitivo a esta industria.[5] [6]

2.3 La llegada de los robots.

El término “Robot” fue acuñado por el escritor checo Karel Capek, quien lo utilizó en 1921 en su drama teatral fantástico *Los robots universales de Rossum (R.U.R.).* El robot nació como un obrero artificial mecánico (del checo *robota*, que quiere decir trabajo duro o trabajo forzoso). Desde ese momento, la palabra “robot” ha sido adoptada universalmente. Por tanto, los robots nacieron en un primer momento como sujetos fantásticos y, más tarde, se utilizó el término para designar genéricamente a cualquier sujeto mecánico que realiza acciones complejas. Así que, también se denomina robots a algunos sistemas de relojería.

Figura 2.3-1 *Robot que aparece en la adaptación checa de 1930 y portada de una edición actual de la obra de Karel Capek Rossum's Universal Robots.*

Lo que distingue a un robot de un mecanismo mecánico genérico no robótico es su capacidad para ser programado. Los robots tienen “tareas” que desempeñar, son programados o son en parte teledirigidos por el hombre para realizar acciones mecánicas específicas. Su segunda característica indispensable es la presencia de un motor que da autonomía de movimiento a los mecanismos. En el pasado, la energía utilizada para el motor podía ser un chorro de agua o de vapor que generaba un empuje o, incluso, un peso que generaba una energía potencial. Más tarde, se pasó a los acumuladores de energía transportables, como los muelles en los sistemas de relojería, hasta llegar a la moderna energía eléctrica, almacenada en baterías y producida por vectores, de los robots electromecánicos. La fusión de los sistemas electrónicos y mecánicos es la mecatrónica, que será la base de los robots antropomorfos del futuro.

Hoy en día, los robots se pueden dividir en las siguientes categorías: fantasía/ ciencia ficción, drones militares, exploración, industria, ciencia/ investigación, medicina, entretenimiento/ juego y relojería.

Los robots abundan en la ciencia ficción, donde nacieron, por lo general, son antropomorfos y casi siempre se utilizan para combatir. El cine ha hecho un uso masivo imaginando androides acabados o en fase de experimentación. Los drones son robots teledirigidos con armas para uso bélico, los más avanzados son los drones volantes dotados de cabezas y sensores. Los robots que realizan exploraciones se utilizan con fines

civiles y van desde las profundidades del mar al espacio sideral (por ejemplo, *Spirit* y *Opportunity*, los dos robots enviados a Marte por la NASA). En la industria, los robots ya han sustituido a los operarios en los trabajos de riesgo y en los repetitivos, por lo general se trata de brazos robotizados. En el campo de la medicina, los encontramos en las prótesis electrónicas de última generación o en los robots quirúrgicos teledirigidos. También el campo del ocio presenta numerosos robots, producidos ya en cantidades industriales como, por ejemplo, el perro AIBO de Sony o el sistema de mecanismos *Lego* con el que se pueden construir robots. Por último, también en los relojes y en los carillones podemos encontrar sujetos robóticos. En la base de todos estos robots subyace la utilización de la mecánica y la electrónica. Leonardo da Vinci, sobre todo en el *Códice Madrid I*, presentó todas las bases mecánicas y los conceptos que se utilizan en los robots modernos: levas, muelles, juntas cardán, transmisión del movimiento, fricción, etc. Se le podría considerar el precursor de la robótica industrial, exceptuando, obviamente, la electrónica, de hecho, uno de los últimos robots avanzados para uso médico ha sido bautizado Da Vinci. [7]

2.4 Automatismos Industriales.

Los actuales sistemas de automatización industrial pueden considerarse como herederos de los autómatas mecánicos del pasado. La definición de autómata que aparece en la real academia índica que un autómata es una "máquina que imita la figura y los movimientos de un ser animado".

La realización física de los automatismos ha dependido continuamente del desarrollo de la tecnología implementándose en primer lugar mediante tecnologías cableadas como la neumática, circuitos de relés electromagnéticos, tarjetas electrónicas. En las dos últimas décadas se han abandonado las tecnologías cableadas sustituidas por los autómatas programables.

Los sistemas de automatización industrial han recibido un gran impulso en este siglo xx sobre todo por parte de la industria del automóvil. El término automatización fue acuñado en 1947 por Delmar S. Halder de la compañía automovilística Ford en Detroit. Halder opina que la *automatización* debería ser un concepto global que abarque todos los diseños y dispositivos realizados para conseguir una plena automatización de la producción. Inicialmente Halder desarrolló su campaña dentro de Ford, pero se extendió por si sola al resto de la industria americana, estableciéndose un debate sobre su aplicación en la industria y las consecuencias sociales que esto conllevaría. Se vertieron opiniones, no sin falta de razón, de que el objetivo final era sacar al ser humano fuera del proceso productivo, prediciendo que una gran cantidad de personas se quedaría sin trabajo.

También se vertieron opiniones favorables dentro del campo tecnológico e industrial, donde muchos consideraban la automatización como un concepto nuevo y revolucionario. La ciencia de la automatización ("Automatology") haría comenzar una nueva era. La automatización supondría "la segunda revolución industrial".

La formalización del tratamiento de los automatismos es muy reciente. Históricamente se puede decir que el tratamiento de los automatismos lógicos se ha basado en el álgebra de Boole y en la teoría de autómatas finitos. No fue hasta la década

de los sesenta que se dispuso de herramientas como las redes de Petri, para el diseño y análisis de automatismos secuenciales y concurrentes.

Tecnologías cableadas:

Las primeras tecnologías disponibles para implementar controladores de sistemas de eventos discretos, se basaban en la aplicación de tecnologías cableadas, lo que de denominaba automatismos cableados. Se utilizaban principalmente las tecnologías neumática y electromecánica.

La tecnología neumática adquiere especial relevancia en la implementación cableada de automatismos, además cuenta con la ventaja de que es homogénea con numerosas máquinas de producción equipadas con cilindros neumáticos. Se debe resaltar que aunque sea una tecnología cableada, el mando neumático utiliza secuenciadores modulares que suprimen una parte del cableado. En la actualidad en muchas máquinas neumáticas industriales el sistema de control que sigue en activo está integrado por circuitos neumáticos. Los nuevos productos desarrollados incorporan como sistema de mando, en el caso de algunas máquinas pequeñas, circuitos de relés electromagnéticos, pero la mayoría esta comandada por autómatas programables.

Los relés electromagnéticos disponen de contactos accionados por una bobina electromagnética. La puesta en tensión de la bobina hace que los contactos conmuten debido a la fuerza electromagnética creada. Los relés electromagnéticos pueden efectuar conmutaciones de grandes corrientes. Continúan siendo interesantes para automatismos muy sencillos. Aunque han sido prácticamente sustituidos por autómatas programables, se siguen utilizando alrededor de ellos en particular para realizar los circuitos de seguridad.

El desarrollo de los autómatas programables:

En las instalaciones de las fábricas de Automóviles se instalaban grandes armarios en paralelo con las líneas de producción. Dentro de estos armarios se construía mediante circuitos de relés electromagnéticos la inteligencia que controlaba el proceso de fabricación. Esta tecnología funcionaba y por supuesto se fabricaban coches pero también poseía una gran problemática.

- La tecnología cableada no era muy adecuada para implementar sistemas de control complejos.
- Los elementos que la forman eran electromecánicos (en el caso de los relés), lo cual implica un número no ilimitado de maniobras (rompen) y la necesidad de implantar logísticas de mantenimiento preventivo.
- Ofrecían una gran dificultad para la búsqueda de averías (un cable que no hace contacto sigue estando visualmente junto al tornillo). Para facilitar la localización de averías se instalaban contactores y relés que señalizarán los fallos.
- A veces se debían realizar conexiones entre cientos o miles de relés, lo que implicaba un enorme esfuerzo de diseño y mantenimiento.
- Cuando se cambiaba el proceso de producción cambiaba también el sistema de control.

Los tiempos de parada ante cualquier avería eran apreciables. Si saltaba una parada de emergencia, se tenía que reiniciar manualmente el sistema, dado que se perdía el estado de la producción.

A finales de los años cincuenta los fabricantes de automóviles necesitaban nuevas y mejores herramientas de control de la producción. Los "nuevos controladores" debían ser fácilmente programables por ingenieros de planta o personal de mantenimiento. El tiempo de vida debía ser largo y los cambios en el programa tenían que realizarse de forma sencilla. Finalmente se imponía que trabajaran sin problemas en entornos industriales adversos. La solución fue el empleo de una técnica de programación familiar y reemplazar los relés mecánicos por relés de estado sólido.

Los autómatas programables se introducen por primera vez en la industria en 1960 aproximadamente. Bedford Associates propuso un sistema de control denominado Controlador Digital Modular (Modicon, Modular Digital Controler) al fabricante de automóviles General Motors.

Otras compañías propusieron a la vez esquemas basados en ordenador, uno de los cuales estaba basado en el PDP-8. El MODICON 084 resultó ser el primer PLC del mundo en ser producido comercialmente.

A mediados de los 70 las tecnologías dominantes de los PLC eran máquinas de estado secuenciales y CPU’s basadas en desplazamiento de bit. Los microprocesadores convencionales incorporaron la potencia necesaria para resolver de forma rápida y completa la lógica de los pequeños PLC's. Por cada modelo de microprocesador había un modelo de PLC basado en el mismo.

Las funciones de comunicación comenzaron a integrarse en los autómatas a partir del año 1973. El primer bus de comunicaciones fue el Modbus de Modicon. El PLC podía ahora establecer comunicación e intercambiar informaciones con otros PLC's.

La implantación de los sistemas de comunicación permitió aplicar herramientas de gestión de producción que se ejecutaban en miniordenadores enviando órdenes de producción a los autómatas de la planta. En las plantas se suele dedicar un autómata programable a ejecutar la función de gestión. Este autómata recibe las órdenes de producción y se encarga de comunicarlas a los autómatas programables dedicados a control. A su vez estos los autómatas de control envían el estado de la producción al autómata de gestión.

En los 80 se produjo un intento de estandarización de las comunicaciones con el protocolo MAP (Manufacturing Automation Protocol) de General Motors. También fue un tiempo en el que se redujeron las dimensiones del PLC y se pasó a programar con programación simbólica a través de ordenadores personales en vez de los clásicos terminales de programación. Hoy día el PLC más pequeño es del tamaño de un simple relé.

En la década de los noventa se ha producido una gradual reducción en el número de nuevos protocolos, y en la modernización de las capas físicas de los protocolos más populares que sobrevivieron a los 80. [8] [9]

Capítulo 3

Base teórica

3.1 Autómatas

3.1.1 **Definición de autómata programable.**

Entendemos por Autómata Programable, o PLC (Controlador Lógico Programable), toda máquina electrónica, diseñada para controlar en tiempo real y en medio industrial procesos secuenciales. Realiza funciones lógicas: series, paralelos, temporizaciones, contajes y otras más potentes como cálculos, regulaciones, etc.

Otra definición de autómata programable sería una «caja» en la que existen, por una parte, unos terminales de entrada (o captadores) a los que se conectan pulsadores, finales de carrera, fotocélulas, detectores...; y por otra, unos terminales de salida (o actuadores) a los que se conectarán bobinas de contactores, electroválvulas, lámparas..., de forma que la actuación de estos últimos está en función de las señales de entrada que estén activas en cada momento, según el programa almacenado.

La función básica de los autómatas programables es la de reducir el trabajo del usuario a realizar el programa, es decir, la relación entre las señales de entrada que se tienen que cumplir para activar cada salida, puesto que los elementos tradicionales (como relés auxiliares, de enclavamiento, temporizadores, contadores...) son internos.

3.1.2 **Estructura de un autómata programable.**

La estructura básica de un autómata programable es la siguiente:

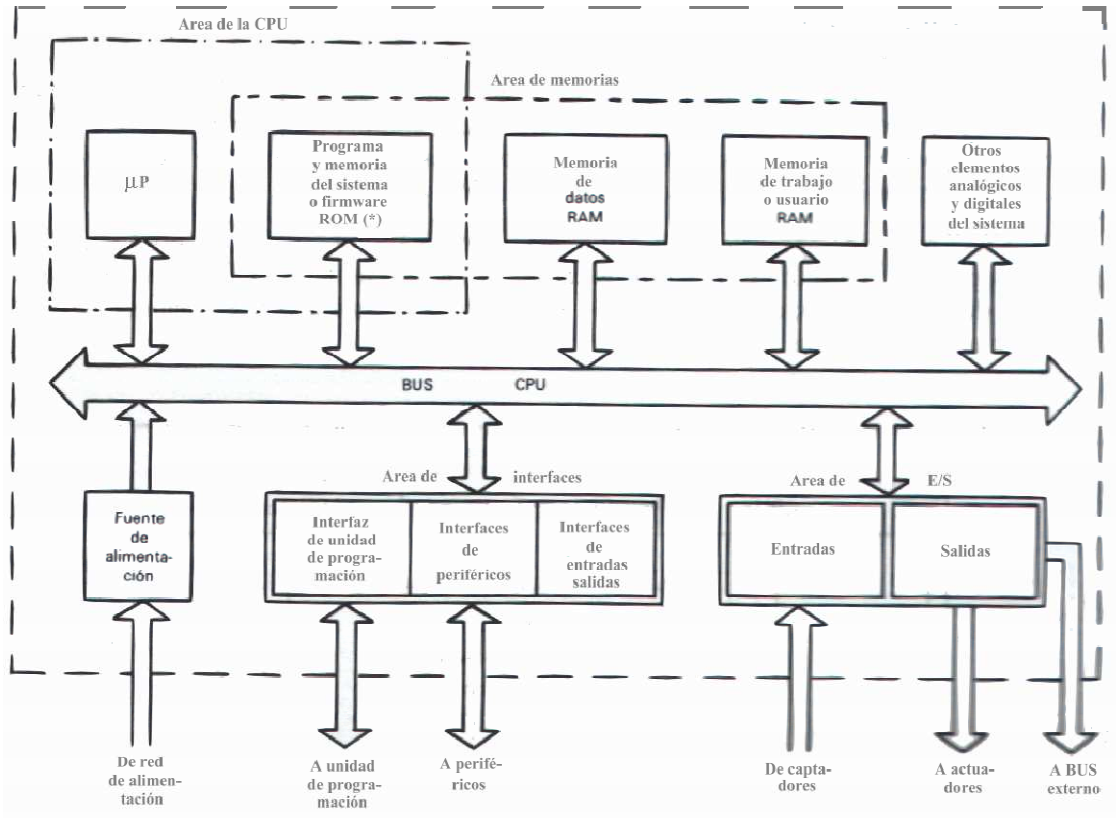


Figura 3.1-1 *Estructura básica de un PLC*

• Fuente de alimentación:

Es la encargada de convertir la tensión de la red, 220v corriente alterna, a baja tensión de corriente continua, normalmente a 24v. Siendo esta la tensión de trabajo en los circuitos electrónicos que forma el Autómata.

• Unidad Central de Procesos o CPU:

Se encarga de recibir las órdenes del operario por medio de la consola de programación y el módulo de entradas. Posteriormente las procesa para enviar respuestas al módulo de salidas. En su memoria se encuentra residente el programa destinado a controlar el proceso. Contiene las siguientes partes:

- Unidad central o de proceso
- Temporizadores y contadores
- Memoria de programa
- Memoria de datos
- Memoria imagen de entrada
- Memoria de salida

• Módulo de entrada:

Es al que se unen los captadores (interruptores, finales de carrera, pulsadores,...).

Cada cierto tiempo el estado de las entradas se transfiere a la memoria imagen de entrada. La información recibida en ella, es enviada a la CPU para ser procesada de acuerdo a la programación.

Se pueden diferenciar dos tipos de captadores conectables al módulo de entradas: los pasivos y los activos.

Los captadores pasivos son los que cambian su estado lógico (activado o no activado) por medio de una acción mecánica. Estos son los interruptores, pulsadores, finales de carrera,...

Los captadores activos son dispositivos electrónicos que suministran una tensión al autómata, que es función de una determinada variable.

• Módulo de salidas:

Es el encargado de activar y desactivar los actuadores (bobinas de contactores, lámparas, motores pequeños,...)

La información enviada por las entradas a la CPU, una vez procesada, se envía a la memoria imagen de salidas, de donde se envía a la interface de salidas para que estas sean activadas y a la vez los actuadores que en ellas están conectados.

Según el tipo de proceso a controlar por el autómata, podemos utilizar diferentes módulos de salidas. Existen tres tipos bien diferenciados:

-A relés: son usados en circuitos de corriente continua y corriente alterna. Están basados en la conmutación mecánica, por la bobina del relé, de un contacto eléctrico normalmente abierto.

- A triac: se utilizan en circuitos de corriente continua y corriente alterna que necesitan maniobras de conmutación muy rápidas.

- A transistores a colector abierto: son utilizados en circuitos que necesiten maniobras de conexión / desconexión muy rápidas. El uso de este tipo de módulos es exclusivo de los circuitos de corriente continua.

• Terminal de programación:

El terminal o consola de programación es el que permite comunicar al operario con el sistema.

Las funciones básicas de éste son las siguientes:

- Transferencia y modificación de programas.

- Verificación de la programación.

- Información del funcionamiento de los procesos.

Como consolas de programación pueden ser utilizadas las construidas específicamente para el autómata, tipo calculadora o bien un ordenador personal, PC, que soporte un software específicamente diseñado para resolver los problemas de programación y control.

• Periféricos:

Los periféricos no intervienen directamente en el funcionamiento del autómata, pero sin embargo facilitan la labor del operario.

Los más utilizados son:

- Grabadoras a cassettes.
- Impresoras.
- Cartuchos de memoria EPROM.
- Visualizadores y paneles de operación OP.
- Memorias EEPROM.

3.1.3 **Forma de funcionamiento del autómata. Concepto de ejecución cíclica.**

La mayoría de los autómatas actuales se basan en el concepto de la ejecución cíclica de las instrucciones ubicadas en su memoria.

El programa es una serie de instrucciones grabadas en la memoria, un ciclo de proceso consiste inicialmente en la consideración de una serie de entradas que seguidamente serán fijadas para todo el ciclo. Después, el autómata ejecuta una instrucción tras otra hasta finalizar el programa y finalmente se definen las ordenes a aplicar sobre las salidas. El ciclo se reproduce así indefinidamente. [11] [12]

3.2 Lenguajes de programación

Cuando surgieron los autómatas programables, lo hicieron con la necesidad de sustituir a los enormes cuadros de maniobra construidos con contactores y relés. Por lo tanto, la comunicación hombre-máquina debería ser similar a la utilizada hasta ese momento. El lenguaje usado, debería ser interpretado, con facilidad, por los mismos técnicos electricistas que anteriormente estaban en contacto con la instalación. Estos lenguajes han evolucionado, en los últimos tiempos, de tal forma que algunos de ellos ya no tienen nada que ver con el típico plano eléctrico a relés.

Se puede definir un programa como un conjunto de instrucciones, órdenes y símbolos reconocibles por el PLC, a través de su unidad de programación, que le permiten ejecutar una secuencia de control deseada. El Lenguaje de Programación en cambio, permite al usuario ingresar un programa de control en la memoria del PLC, usando una sintaxis establecida.

Al igual que los PLCs se han desarrollado y expandido, los lenguajes de programación también se han desarrollado con ellos. Los lenguajes de hoy en día tienen nuevas y más versátiles instrucciones y con mayor poder de computación. Por ejemplo, los PLCs pueden transferir bloques de datos de una localización de memoria a otra, mientras al mismo tiempo llevan cabo operaciones lógicas y matemáticas en otro bloque. Como resultado de estas nuevas y expandidas instrucciones, los programas de control pueden ahora manejar datos más fácilmente.

Adicionalmente a las nuevas instrucciones de programación, el desarrollo de nuevos módulos de entradas y salidas también ha obligado a cambiar las instrucciones existentes.

3.2.1 **Tipos de lenguajes de programación de PLCs:**

En la actualidad cada fabricante diseña su propio software de programación, lo que significa que existe una gran variedad comparable con la cantidad de PLCs que hay en el mercado. No obstante, actualmente existen tres tipos de lenguajes de programación de PLCs como los más difundidos a nivel mundial, estos son:

- Lenguaje de contactos o Ladder
- Lenguaje Booleano (Lista de instrucciones)
- Diagrama de funciones

Es obvio, que la gran diversidad de lenguajes de programación da lugar a que cada fabricante tenga su propia representación, originando cierta incomodidad al usuario cuando programa más de un PLC.

La Comisión Electrotécnica Internacional (IEC) desarrolló el estándar IEC 1131, en un esfuerzo por estandarizar los Controladores Programables. Uno de los objetivos del Comité fue crear un conjunto común de instrucciones que podría ser usado en todos los PLCs. Aunque el estándar 1131 alcanzó el estado de estándar internacional en agosto de 1992, el esfuerzo para crear un PLC estándar global ha sido una tarea muy difícil debido a la diversidad de fabricantes de PLCs y a los problemas de incompatibilidad de programas entre marcas de PLCs.

El estándar IEC 1131 para controladores programables consiste de cinco partes, una de las cuales hace referencia a los lenguajes de programación y es referida como la IEC 1131-3.

El estándar IEC 1131-3 define dos lenguajes gráficos y dos lenguajes basados en texto, para la programación de PLCs. Los lenguajes gráficos utilizan símbolos para programar las instrucciones de control, mientras los lenguajes basados en texto, usan cadenas de caracteres para programar las instrucciones.

- **Lenguajes Gráficos**
  - Diagrama Ladder (LD)
  - Diagrama de Bloques de Funciones (FBD)
- **Lenguajes Textuales**
  - Lista de Instrucciones (IL)
  - Texto Estructurado (ST)

Adicionalmente, el estándar IEC 1131-3 incluye una forma de programación orientada a objetos llamada Sequential Function Chart (SFC). SFC es a menudo categorizado como un lenguaje IEC 1131-3, pero éste es realmente una estructura organizacional que coordina los cuatro lenguajes estándares de programación (LD, FBD, IL y ST). La estructura del SFC tuvo sus raíces en el primer estándar francés de Grafcet (IEC 848).

3.2.2 **Lenguaje Ladder (LD):**

El ladder, también denominado lenguaje de contactos o de escalera, es un lenguaje de programación gráfico muy popular dentro de los Controladores Lógicos Programables (PLC), debido a que es el que más similitudes tiene con el utilizado por un electricista al elaborar cuadros de automatismos. Muchos autómatas incluyen módulos especiales de software para poder programar gráficamente de esta forma.

Elementos de programación:

Para programar un PLC con LADDER, además de estar familiarizado con las reglas de los circuitos de conmutación, es necesario conocer cada uno de los elementos de que consta este lenguaje. En la siguiente tabla se pueden observar los símbolos de los elementos básicos junto con sus respectivas descripciones.

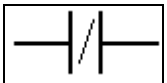
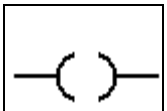
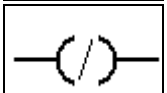
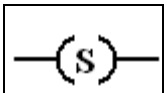
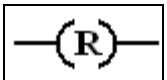
Símbolo	Nombre	Descripción
	Contacto NA	Se activa cuando hay un uno lógico en el elemento que representa, esto es, una entrada (para captar información del proceso a controlar), una variable interna o un bit de sistema.
	Contacto NC	Su función es similar al contacto NA anterior, pero en este caso se activa cuando hay un cero lógico, cosa que deberá de tenerse muy en cuenta a la hora de su utilización.
	Bobina NA	Se activa cuando la combinación que hay a su entrada (izquierda) da un uno lógico. Su activación equivale a decir que tiene un uno lógico. Suele representar elementos de salida, aunque a veces puede hacer el papel de variable interna.
	Bobina NC	Se activa cuando la combinación que hay a su entrada (izquierda) da un cero lógico. Su activación equivale a decir que tiene un cero lógico. Su comportamiento es complementario al de la bobina NA.
	Bobina SET	Una vez activa (puesta a 1) no se puede desactivar (puesta a 0) si no es por su correspondiente bobina en RESET. Sirve para memorizar bits y usada junto con la bina RESET dan una enorme potencia en la programación.
	Bobina SET	Permite desactivar una bobina SET previamente activada.

Tabla 3.2-1 *Elementos de programacióndel lenguaje de contactos*

Programación:

Una vez conocidos los elementos que LADDER proporciona para su programación, resulta importante resaltar cómo se estructura un programa y cuál es el orden de ejecución.

El siguiente esquema representa la estructura general de la distribución de todo programa LADDER, contactos a la izquierda y bobinas y otros elementos a la derecha.

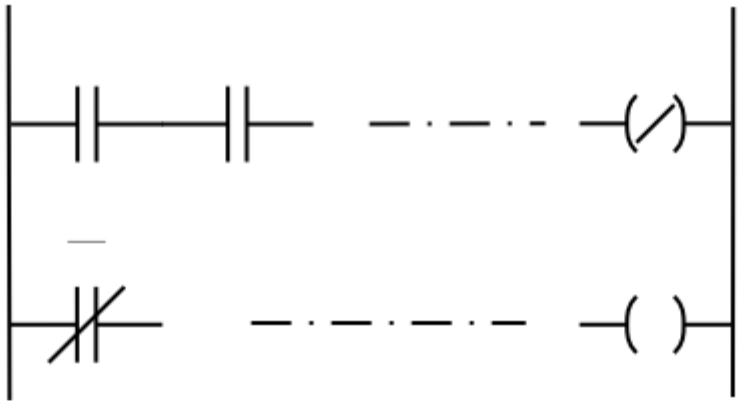


Figura 3.2-1 *Esquema de programación Ladder*

En cuanto a su equivalencia eléctrica, es fácil imaginar que las líneas verticales representan las líneas de alimentación de un circuito de control eléctrico.

El orden de ejecución es generalmente de arriba hacia abajo y de izquierda a derecha, primero los contactos y luego las bobinas, de manera que al llegar a éstas ya se conoce el valor de los contactos y se activan si procede. El orden de ejecución puede variar de un controlador a otro, pero siempre se respetará el orden de introducción del programa, de manera que se ejecuta primero lo que primero se introduce.

3.2.3 **Diagrama de funciones (FBD):**

Es un lenguaje gráfico que permite al usuario programar elementos (bloque de funciones del PLC) en tal forma que ellos aparecen interconectados al igual que un circuito eléctrico. Generalmente utilizan símbolos lógicos para representar al bloque de función. Las salidas lógicas no requieren incorporar una bobina de salida, porque la salida es representada por una variable asignada a la salida del bloque.

El diagrama de funciones lógicas, resulta especialmente cómodo de utilizar, a técnicos habituados a trabajar con circuitos de puertas lógicas, ya que la simbología usada en ambos es equivalente.

Adicionalmente a las funciones lógicas estándares y específicas del vendedor, el lenguaje FBD de la Norma IEC 1131-3 permite al usuario construir sus propios bloques de funciones, de acuerdo a los requerimientos del programa de control.

Ejemplo de programación mediante diagrama de funciones:

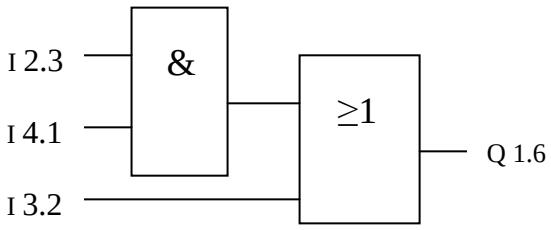


Figura 3.2-2 *Esquema de programación con diagrama de bloques*

3.2.4 **Lenguaje booleano o lista de instrucciones (IL):**

El lenguaje Booleano utiliza la sintaxis del Álgebra de Boole para ingresar y explicar la lógica de control.

En los autómatas de gama baja, es el único modo de programación. Consiste en elaborar una lista de instrucciones o nemónicos, haciendo uso de operadores Booleanos (AND, OR, NOT, etc.) y otras instrucciones nemónicas, para implementar el circuito de control. El lenguaje “Lista de Instrucciones” (IL) de la Norma IEC 1131-3, es una forma de lenguaje Booleano.

También decir, que este tipo de lenguaje es, en algunos casos, la forma más rápida de programación e incluso la más potente.

Ejemplo de programación Booleana:

```
A   I     2.3
A   I     4.1
O   I     3.2
=   Q     1.6
```

Figura 3.2-3 *Esquema de programación booleana*

3.2.5 **Lenguaje de texto estructurado (ST):**

Texto estructurado (ST) es un lenguaje de alto nivel que permite la programación estructurada, lo que significa que muchas tareas complejas pueden ser divididas en unidades más pequeñas. ST se parece mucho a los lenguajes de computadoras BASIC o PASCAL, que usa subrutinas para llevar a cabo diferentes partes de las funciones de control y paso de parámetros y valores entre las diferentes secciones del programa.

Al igual que LD, FBD e IL, el lenguaje de texto estructurado utiliza la definición de variables para identificar entradas y salidas de dispositivos de campo y cualquier otra variable creada internamente.

Incluye estructuras de cálculo repetitivo y condicional, tales como: FOR ... TO; REPEAT..... UNTIL X; WHILE X... ; IF ... THEN ...ELSE. Además soporta operaciones Booleanas (AND, OR, etc.) y una variedad de datos específicos, tales como fecha, hora.

La programación en Texto Estructurado es apropiada para aplicaciones que involucran manipulación de datos, ordenamiento computacional y aplicaciones matemáticas que utilizan valores de punto flotante. ST es el mejor lenguaje para la implementación de aplicaciones de inteligencia artificial, lógica difusa, toma de decisiones, etc.

Ejemplo:

```
IF value < 7 THEN
        WHILE value < 8 DO
                value := value + 1;
        END_WHILE;
END_IF;
```

Figura 3.2-4 *Esquema de programación en Texto Estructurado*

3.2.6 **Diagrama de flujo secuencial (SFC):**

Es un “lenguaje” gráfico que provee una representación diagramática de secuencias de control en un programa. Básicamente, SFC es similar a un diagrama de flujo, en el que se puede organizar los subprogramas o subrutinas (programadas en LD, FBD, IL y/o ST) que forman el programa de control. SFC es particularmente útil para operaciones de control secuencial, donde un programa fluye de un punto a otro una vez que una condición ha sido satisfecha (cierta o falsa).

El marco de programación de SFC contiene tres principales elementos que organizan el programa de control:

- Pasos (etapas)
- Transiciones (condiciones)
- Acciones

El programa irá activando cada una de las etapas y desactivando la anterior conforme se vayan cumpliendo cada una de las condiciones. Las acciones se realizarán en función de la etapa activa a la que están asociadas. Por ejemplo, la etapa 1 activa tras arrancar el programa, al cumplirse la "Condición 1", se activará la etapa 2, se desactivará la 1, y se realizará la "Acción 1".

Ejemplo:

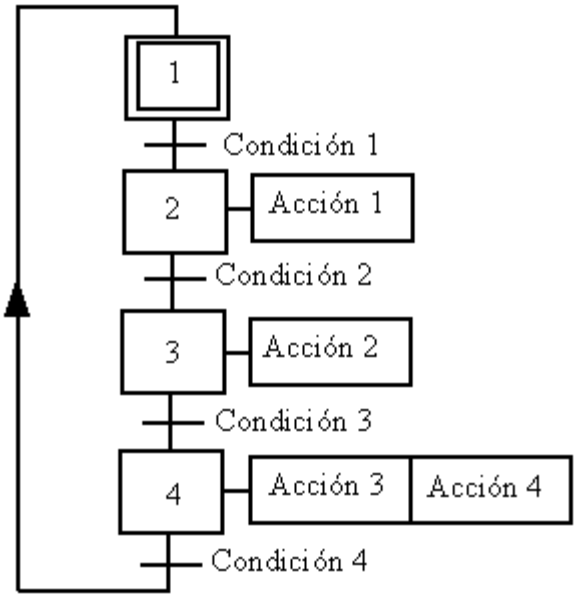


Figura 3.2-5 *Esquema de programación en diagrama de flujo secuencial*

El lenguaje SFC tiene su origen en el estándar francés GRAFCET (GRAFica de Control de Etapas de Transición). El grafcet también utiliza etapas, transiciones y acciones, que operan de la misma manera como en SFC.

Fue especialmente diseñado para resolver problemas de automatismos secuenciales. Las acciones son asociadas a las etapas y las condiciones a cumplir a las transiciones. Este lenguaje resulta enormemente sencillo de interpretar por operarios sin conocimientos de automatismos eléctricos. Muchos de los autómatas que existen en el mercado permiten la programación en GRAFCET, tanto en modo gráfico o como por lista de instrucciones. También se puede utilizar para resolver problemas de automatización de forma teórica y posteriormente convertirlo a plano de contactos.[11]

3.3 SCADA y HMI

El control y la automatización de los procesos industriales han motivado numerosos desarrollos en el campo tecnológico. Uno de los más importantes ha sido el Sistema de Supervisión, que permite inspeccionar, configurar, hacer un seguimiento de las variables y, tomar acciones correctivas en tiempo real de un proceso, directamente desde la planta o desde una central remota ubicada incluso a cientos de kilómteros de ésta, lo que los hace imprescindibles dentro de los sistemas de control actuales.

Un Sistema de Supervisión permite conocer los diferentes estados de las variables que forman parte de un proceso supervisado. Su configuración se realiza por medio de las Aplicaciones de Supervisión HMI/SCADA, que son programas orientados a objetos, con un amplio manejo de la información, una buena interfaz gráfica y un gran número de funciones y posibilidades de comunicación.

3.3.1 **Interfaz Humano-Máquina.**

Una interfaz Hombre - Máquina o HMI ("Human Machine Interface") es el aparato que presenta los datos a un operador (humano) y a través del cual éste controla el proceso.

Los sistemas HMI pueden definirse como una "ventana de un proceso". Esta ventana puede estar en dispositivos especiales como paneles de operador o en un ordenador. Los sistemas HMI en ordenadores se los conoce también como software (o aplicación) HMI o de monitorización y control de supervisión. Las señales del proceso son conducidas al HMI por medio de dispositivos como tarjetas de entrada/salida en el ordenador, PLC's (Controladores lógicos programables), PACs (Controlador de automatización programable), RTU (Unidades remotas de I/O) o DRIVER's (Variadores de velocidad de motores). Todos estos dispositivos deben tener una comunicación que entienda el HMI.

La industria de HMI nació esencialmente de la necesidad de estandarizar la manera de monitorizar y de controlar múltiples sistemas remotos, PLCs y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA (es el acrónimo de *Supervisory Control And Data Acquisition*; Supervisión, Control y Adquisición de Datos) lo hacen de manera automática. Históricamente los PLC no tienen una manera estándar de presentar la información al operador.

Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de problemas. Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLCs.

SCADA es popular debido a esta compatibilidad y seguridad. Ésta se usa desde aplicaciones a pequeñas escalas, como controladores de temperatura en un espacio, hasta aplicaciones muy grandes como el control de plantas nucleares.

3.3.2 **Componentes del sistema**

Los tres componentes de un sistema SCADA son:

- Unidades de Terminal Remota (también conocida como UTR, RTU o Estaciones Externas).
- Estación Maestra y Computador con HMI.
- Infraestructura de Comunicación.

Unidad de Terminal Remota (RTU).

La RTU se conecta al equipo físicamente y lee los datos de estado como los estados abierto/cerrado desde una válvula o un interruptor, lee las medidas como presión, flujo,

voltaje o corriente. Por el equipo el RTU puede enviar señales que pueden controlarlo: abrirlo, cerrarlo, intercambiar la válvula o configurar la velocidad de la bomba, ponerla en marcha, pararla.

La RTU puede leer el estado de los datos digitales o medidas de datos analógicos y envía comandos digitales de salida o puntos de ajuste analógicos.

Una de las partes más importantes de la implementación de SCADA son las alarmas. Una alarma es un punto de estado digital que tiene cada valor NORMAL o ALARMA. La alarma se puede crear en cada paso que los requerimientos lo necesiten. Un ejemplo de un alarma es la luz de "tanque de combustible vacío" del automóvil. El operador de SCADA pone atención a la parte del sistema que lo requiera, por la alarma. Pueden enviarse por correo electrónico o mensajes de texto con la activación de una alarma, alertando al administrador o incluso al operador de SCADA.

Estación Maestra.

El termino "Estación Maestra" se refiere a los servidores y al software responsable para comunicarse con el equipo del campo (RTUs, PLCs, etc) en estos se encuentra el software HMI corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador, A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores y aplicaciones de software distribuido.

El sistema SCADA usualmente presenta la información al personal operativo de manera gráfica, en forma de un diagrama de representación. Esto significa que el operador puede ver un esquema que representa la planta que está siendo controlada. Por ejemplo un dibujo de una bomba conectada a la tubería puede mostrar al operador cuanto fluido está siendo bombeado desde la bomba a través de la tubería en un momento dado o bien el nivel de líquido de un tanque o si la válvula está abierta o cerrada. Los diagramas de representación puede consistir en gráficos de líneas y símbolos esquemáticos para representar los elementos del proceso, o pueden consistir en fotografías digitales de los equipos sobre los cuales se animan las secuencias.

Los bloques software de un SCADA (módulos), permiten actividades de adquisición, supervisión y control

3.3.3 **Funciones**

Dentro de las funciones básicas realizadas por un sistema SCADA están las siguientes:

- Recabar, almacenar y mostrar información, en forma continua y confiable, correspondiente a la señalización de campo: estados de dispositivos, mediciones, alarmas, etc.

- Ejecutar acciones de control iniciadas por el operador, tales como: abrir o cerrar válvulas, arrancar o parar bombas, etc.

- Alertar al operador de cambios detectados en la planta, tanto aquellos que no se consideren normales (alarmas) como cambios que se produzcan en la operación diaria de la planta (eventos). Estos cambios son almacenados en el sistema para su posterior análisis.

- Aplicaciones en general, basadas en la información obtenida por el sistema, tales como: reportes, gráficos de tendencia, historia de variables, cálculos, predicciones, detección de fugas, etc.[11]

3.4 Variación de velocidad

Las aplicaciones industriales y domésticas en las que se necesita una variación de velocidad continua, son cada vez más frecuentes. La regulación de velocidad puede realizarse por métodos mécanicos, como poleas o engranajes, o por medios eléctricos.

Los motores de corriente alterna son los más comunes en aplicaciones industriales debido al casi inexistente mantenimiento, y a que estos motores abarcan el mayor margen de potencias.

Para garantizar un funcionamiento seguro, no se pueden conectar directamente a la red, serán necesarios diversos dsipositivos, entre los que se encuentra el variador de velocidad.

En la fase de arranque de estos motores, el par debe ser el necesario para mover la carga con una aceleración adecuada hasta que se alcanza la velocidad de funcionamiento en régimen permanente, procurando que no aparezcan problemas eléctricos o mecánicos capaces de perjudicar al motor, a la instalación eléctrica o a los elementos que hay que mover.

El motor de corriente alterna, a pesar de ser un motor robusto, de poco mantenimiento, liviano e ideal para la mayoría de las aplicaciones industriales, tiene el inconveniente de ser un motor rígido en cuanto a su velocidad. La velocidad del motor asíncrono depende de la forma constructiva del motor y de la frecuencia de alimentación. Como la frecuencia de alimentación que entregan las Compañías de electricidad es constante, la velocidad de los motores asíncronos es constante, salvo que se varíe el número de polos, el deslizamiento o la frecuencia.

El método más eficiente de controlar la velocidad de un motor eléctrico es por medio de un variador electrónico de frecuencia. No se requieren motores especiales, son mucho más eficientes y tienen precios cada vez más competitivos.

El variador de frecuencia regula la frecuencia del voltaje aplicado al motor, logrando modificar su velocidad. Sin embargo, simultáneamente con el cambio de frecuencia, debe variarse el voltaje aplicado al motor para evitar la saturación del flujo magnético con una elevación de la corriente que dañaría el motor.

3.4.1 **Descripción**

Los variadores son convertidores de energía encargados de modular la energía que recibe el motor. Estos dispositivos convierten las magnitudes fijas de frecuencia y tensión de red, en magnitudes variables.

Se puede ver una comparación de las características de funcionamiento, utilizando un motor asíncrono normal y uno con variador de velocidad para entender algunas de las ventajas que ofrecen estos dispositivos.

Motor asíncrono	**En uso normal**	**Con variador de velocidad**
Corriente de arranque	Muy elevada, del orden de 6-8 veces la corriente nominal en valor eficaz, 15-20 veces en valor cresta	Limitado en el motor (en general, cerca de 1,5 veces la corriente nominal)
Par de arranque Cd	Elevado y no controlado, del orden de 2 a 3 veces el par nominal Cn	Del orden de 1,5 veces el par nominal Cn y controlado durante toda la aceleración
Arranque	Brutal, cuya duración sólo depende de las características del motor y de la carga arrastrada (Par resistente, inercia)	Progresivo, sin brusquedades y controlado (rampa lineal de velocidad, por ejemplo)
Velocidad	Variando ligeramente según la carga (próxima a la velocidad de sincronismo Ns)	Variación posible a partir de cero hasta un valor superior a la velocidad de sincronismo Ns
Par máximo Cm	Elevado, del orden de 2-3 veces el par nominal Cn	Elevado disponible para todo el rango de velocidades (del orden de 1,5 veces el par nominal)
Frenado eléctrico	Relativamente complejo, necesita protecciones y un esquema particular	Fácil
Inversión del sentido de marcha	Fácil solamente después de la parada motor	Fácil
Riesgo de bloqueo	Si, en caso de exceso de par (par resistente > Cm), o en caso de bajada de tensión	No
Funcionamiento del motor en el plano par-velocidad	Gráfico 1	Gráfico 2

Tabla 3.4-1 *Comparación de las características de funcionamiento que demuestran el gran interés de los variadores de velocidad.*

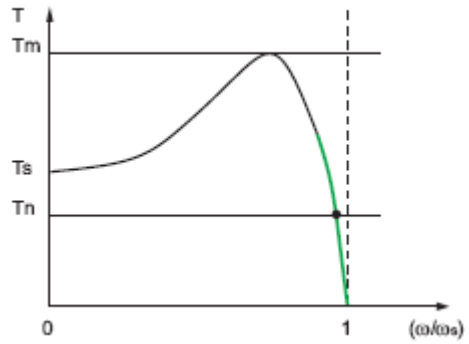


Figura 3.4-1 *Diagrama par-velocidad de un motor alimentado en directo. La zona de funcionamiento del motor en el plano par-velocidad está limitada a la parte verde de la curva.*

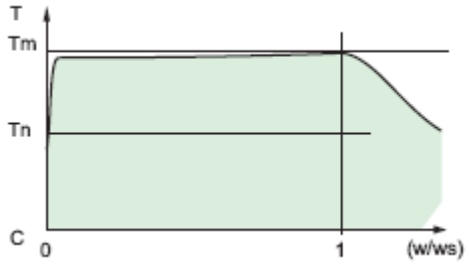


Figura 3.4-2 *Diagrama par-velocidad de un motor alimentado por convertidor de frecuencia. Aquí la zona de funcionamiento del motor en el plano par-velocidad está representada en verde.*

Composición

Los arrancadores y variadores de velocidad electrónicos se componen de dos módulos generalmente montados en una misma envolvente:

- Un módulo de control que controla el funcionamiento del aparato,

- Un módulo de potencia que alimenta el motor con energía eléctrica.

El módulo de control

En los arrancadores y variadores modernos, todas las funciones se controlan mediante un microprocesador que gestiona la configuración, las órdenes transmitidas por un operador o por una unidad de proceso y los datos proporcionados por las medidas como la velocidad, la corriente, etcétera.

Las capacidades de cálculo de los microprocesadores, así como de los circuitos dedicados (ASIC) han permitido diseñar algoritmos de mando con excelentes prestaciones y en particular, el reconocimiento de los parámetros de la máquina arrastrada. A partir de estas informaciones, el microprocesador gestiona las rampas de aceleración y deceleración, el control de la velocidad y la limitación de corriente, generando las señales de control de los componentes de potencia. Las protecciones y la seguridad son procesadas por circuitos especializados (ASIC) o están integradas en los módulos de potencia (IPM).

Los límites de velocidad, las rampas, los límites de corriente y otros datos de configuración, se definen usando un teclado integrado o mediante PLC (sobre buses de campo) o mediante PC.

Del mismo modo, los diferentes comandos (marcha, parada, frenado...) pueden proporcionarse desde interfaces de diálogo hombre/máquina, utilizando autómatas programables o PC.

Los parámetros de funcionamiento y las informaciones de alarma, y los defectos pueden verse mediante displays, diodos LED, visualizadores de segmentos o de cristal líquido o pueden enviarse hacia la supervisión mediante un bus de campo.

Los relés, frecuentemente programables, proporcionan información de:

- fallos (de red, térmicos, de producto, de secuencia, de sobrecarga),

- vigilancia (umbral de velocidad, pre-alarma, final de arranque).

Las tensiones necesarias para el conjunto de circuitos de medida y de control son proporcionadas por una alimentación integrada en el variador y separadas galvánicamente de la red.

El módulo de potencia

El módulo de potencia está principalmente constituido por:

- Componentes de potencia (diodos, tiristores, IGBT...),
- Interfaces de medida de las tensiones y/o corrientes,
- Frecuentemente de un sistema de ventilación.

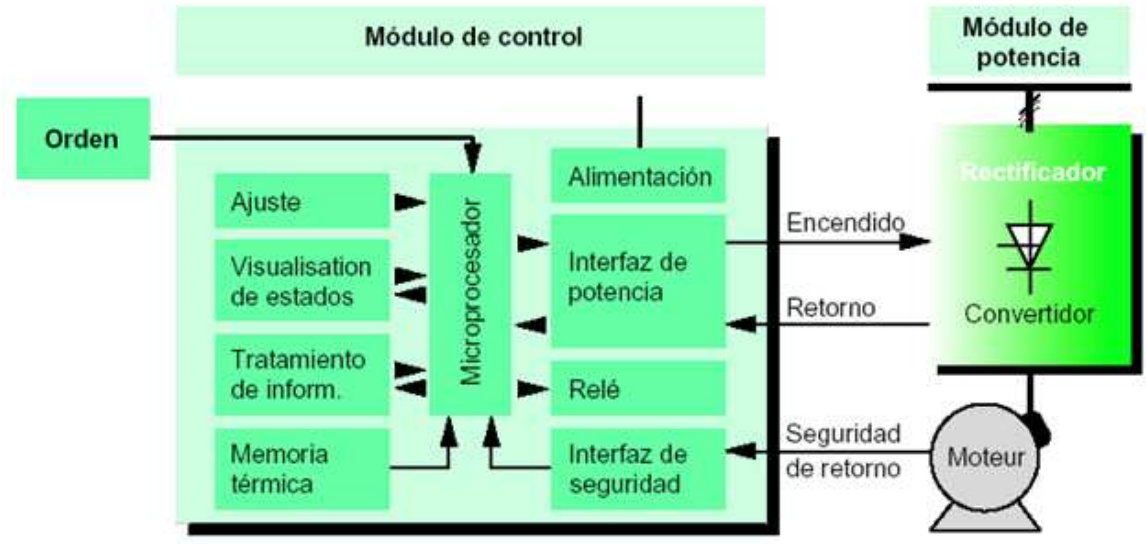


Figura 3.4-3 *Estructura general de un variador de velocidad electrónico*

Los componentes de potencia (Figura 3.4-4 *Componentes de potencia*) son semiconductores que funcionan en «todo o nada», comparables, por tanto, a los interruptores estáticos que pueden tomar dos estados: abierto o cerrado.

Estos componentes, integrados en un módulo de potencia, constituyen un convertidor que alimenta, a partir de la red a tensión y frecuencia fijas, un motor eléctrico con una tensión y/o frecuencia variables.

Los componentes de potencia son la clave de la variación de velocidad y los progresos realizados estos últimos años han permitido la fabricación de variadores de velocidad económicos.

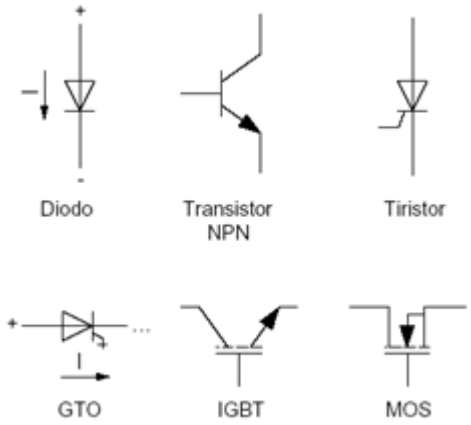


Figura 3.4-4 *Componentes de potencia*

3.4.2 **Modos de funcionamiento**

Funcionamiento a par constante

Se denomina funcionamiento a par constante cuando las características de la carga son tales, que, en régimen permanente, el par solicitado es sensiblemente constante sea cual sea la velocidad (Figura 3.4-5 *Curva de funcionamiento a par constante*). Este modo de funcionamiento se utiliza en las cintas transportadoras y en las amasadoras. Para este tipo de aplicaciones, el variador debe tener la capacidad de proporcionar un par de arranque importante (1,5 veces o más el par nominal) para vencer los rozamientos estáticos y para acelerar la máquina (inercia).

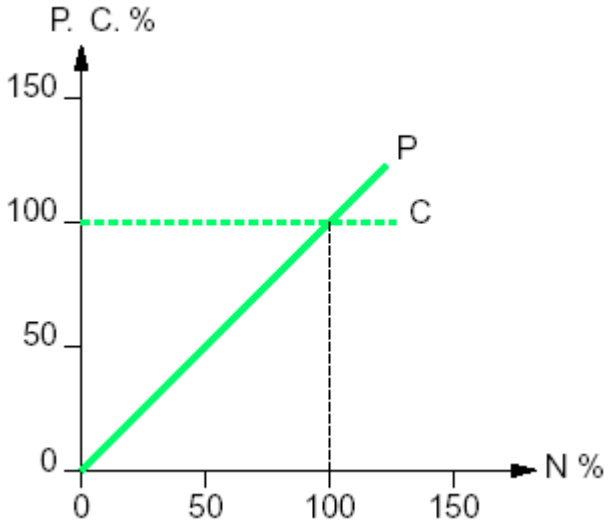


Figura 3.4-5 *Curva de funcionamiento a par constante*

Funcionamiento a par variable

Se denomina funcionamiento a par variable cuando las características de la carga son tales que en régimen permanente, el par solicitado varía con la velocidad. Es en concreto el caso de las bombas volumétricas con tornillo de Arquímedes cuyo par crece

linealmente con la velocidad (Figura 3.4-6 *Curva de funcionamiento a par variable*) o las máquinas centrífugas (bombas y ventiladores) cuyo par varía con el cuadrado de la velocidad (Figura 3.4-6 *Curva de funcionamiento a par variable*).

Para un variador destinado a este tipo de aplicaciones, es suficiente un par de arranque mucho menor (en general 1,2 veces el par nominal del motor). Muy frecuentemente dispone de funciones complementarias como la posibilidad de omitir las frecuencias de resonancia correspondientes a las vibraciones indeseables de la máquina. Es imposible funcionar más allá de la frecuencia nominal de la máquina porque sería una carga insoportable para el motor y el variador.

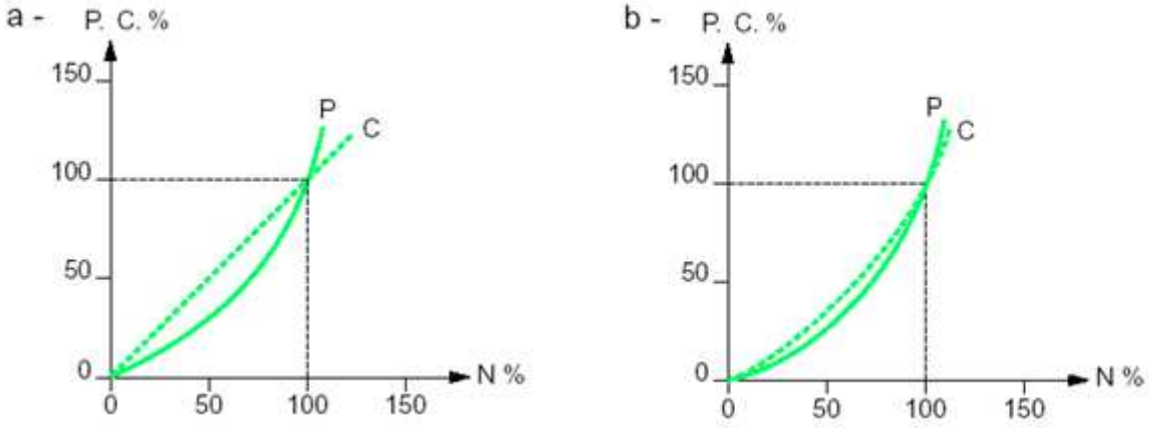


Figura 3.4-6 *Curva de funcionamiento a par variable*

3.4.3 **Ventajas e Inconvenientes**

Ventajas de la utilización del Variador de Velocidad en el arranque de motores asíncronos.

- El variador de velocidad no tiene elementos móviles, ni contactos.
- La conexión del cableado es muy sencilla.
- Permite arranques suaves, progresivos y sin saltos.
- Controla la aceleración y el frenado progresivo.
- Limita la corriente de arranque.
- Permite el control de rampas de aceleración y deceleración regulables en el tiempo.
- Consigue un ahorro de energía cuando el motor funcione parcialmente cargado, con acción directa sobre el factor de potencia
- Puede detectar y controlar la falta de fase a la entrada y salida de un equipo. Protege al motor.
- Puede controlarse directamente a través de un autómata o microprocesador.
- Se obtiene un mayor rendimiento del motor.

- Permite ver las variables (tensión, frecuencia, r.p.m, etc…).

Inconvenientes de la utilización del Variador de Velocidad en el arranque de motores asíncronos.

- Es un sistema caro, pero rentable a largo plazo.
- Requiere estudio de las especificaciones del fabricante.
- Requiere un tiempo para realizar la programación.[13]

3.5 Comunicaciones

En el mundo industrial los equipos tienen orígenes y funciones muy heterogéneas. Se necesita de esa heterogeneidad para optimizar costes y cubrir las necesidades, en continua evolución, propias de la producción.

Los mecanismos que permiten dar coherencia a esa heterogeneidad se basan en el diálogo entre los distintos sistemas, para ello se han creado las diferentes posibilidades de comunicación.

En el siguiente gráfico se puede ver cuáles son en gran medida las comunicaciones exitentes en el mercado en la actualidad ordenadas según su aplicación y sus propiedades.

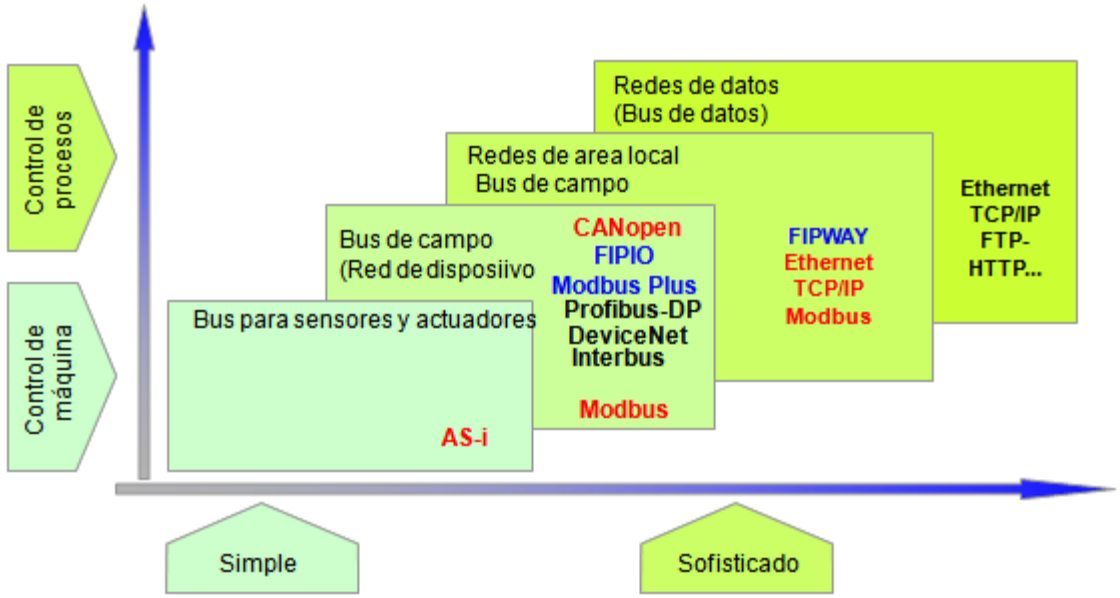


Figura 3.5-1 *Principales redes y buses*

Para este proyecto en concreto, se estaría hablando de un control a nivel de máquina, no llega a ser un control de procesos como puede ser un fábrica, por lo que conviene centrarse en el estudio de los buses de campo, más concretamente en Modbus y CANopen.

3.5.1 **Modbus**

Modbus es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI (Unidad de datos en capa de aplicación), basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar, en la industria es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales.

El protocolo MODBUS define una estructura de mensajes que puede ser reconocida por diferentes dispositivos independientemente del tipo de red de comunicaciones utilizada. El protocolo describe el proceso para acceder a información de un dispositivo, cómo debe responder éste, y como se notifican las situaciones de error. El protocolo MODBUS define una red digital de comunicaciones con un solo maestro y uno o más dispositivos esclavo.

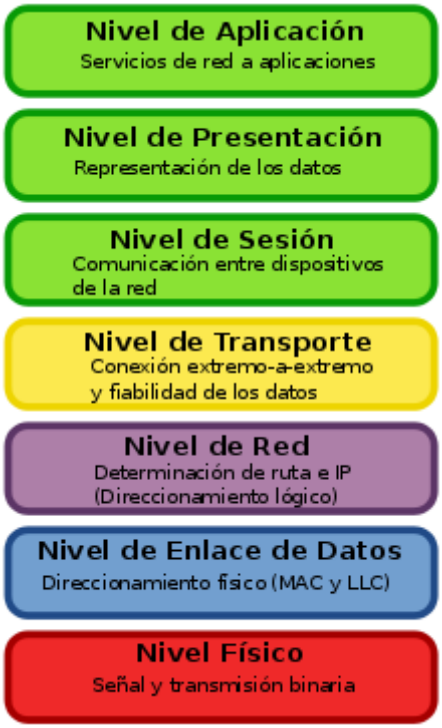


Figura 3.5-2 *Capas del modelo OSI*

Las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son:

- es público
- su implementación es fácil y requiere poco desarrollo
- maneja bloques de datos sin suponer restricciones

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA). Existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP).

Existen dos modos de transmisión, con diferentes representaciones numéricas de los datos y detalles del protocolo ligeramente desiguales. Modbus RTU (Remote Terminal Unit) es una representación binaria compacta de los datos. Modbus ASCII (American Satandard Code for Information Interchange) es una representación legible del protocolo pero menos eficiente. Ambas implementaciones del protocolo son serie. El formato RTU finaliza la trama con una suma de control de redundancia cíclica (CRC), mientras que el formato ASCII utiliza una suma de control de redundancia longitudinal (LRC).

En una red de dispositivos conectados mediante el protocolo MODBUS no se pueden compartir dispositivos utilizando diferentes modos de transmisión.

La versión Modbus/TCP es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP.

Cada dispositivo de la red Modbus posee una dirección única. Cualquier dispositivo puede enviar órdenes Modbus, aunque lo habitual es permitirlo sólo a un dispositivo maestro. Cada comando Modbus contiene la dirección del dispositivo destinatario de la orden. Todos los dispositivos reciben la trama pero sólo el destinatario la ejecuta (salvo un modo especial denominado "Broadcast"). Cada uno de los mensajes incluye información redundante que asegura su integridad en la recepción. Los comandos básicos Modbus permiten controlar un dispositivo RTU para modificar el valor de alguno de sus registros o bien solicitar el contenido de dichos registros.

La mayoría de problemas presentados hacen referencia a la latencia y a la sincronización.

Estructura del mensaje:

Un mensaje consiste en una secuencia de caracteres que puedan ser interpretados por el receptor. Esta secuencia de caracteres define la trama.

Para sincronizar la trama, los dispositivos receptores monitorizan el intervalo de tiempo transcurrido entre caracteres recibidos. Si se detecta un intervalo mayor que tres veces y media el tiempo necesario para transmitir un carácter, el dispositivo receptor ignora la trama y asume que el siguiente carácter que recibirá será una dirección.

3,5T	DIRECCIÓN	FUNCIÓN	DATOS	CRC	3,5T
3,5 bytes	1 byte	1 byte	N bytes	2 bytes	3,5 bytes

Tabla 3.5-1 *Estructura de un mensaje Modbus*

- 1 Dirección

El campo dirección es el primero de la trama después del tiempo de sincronización. Indica el dispositivo al que va dirigido el mensaje. Cada dispositivo de la red debe tener asignada una dirección única, diferente de cero. Igualmente, cuando un dispositivo responde a un mensaje, debe enviar en primer lugar su dirección para que el maestro reconozca la procedencia del mensaje.

MODBUS permite enviar mensajes a todos los dispositivos a la vez utilizando para ello la dirección cero.

- 2 Función

El campo función indica al dispositivo direccionado qué tipo de función ha de realizar.

- 3 Datos

El campo datos contiene la información necesaria para que los dispositivos puedan ejecutar las funciones solicitadas, o la información enviada por los dispositivos al maestro como respuesta a una función.

- 4 CRC

El campo CRC es el último de la trama y permite al maestro y a los dispositivos detectar errores de transmisión. Ocasionalmente, debido a ruido eléctrico o a interferencias de otra naturaleza, se puede producir alguna modificación en el mensaje mientras se está transmitiendo. El control de errores por medio de CRC asegura que los dispositivos receptores o el maestro no efectuaran acciones incorrectas debido a una modificación accidental del mensaje. [14]

3.5.2 **CANopen**

El protocolo CAN es un estándar que viene descrito en el estándar ISO 11898, inicialmente impulsado por el fabricante alemán BOSCH para simplificar el cableado en los automóviles Mercedes-Benz. Así, una gran cantidad de aplicaciones dónde se utiliza más ampliamente es en automoción, donde existe gran cantidad de electrónica asociada a los elementos instalados tanto en el motor como en el resto del vehículo (airbag, cinturones de seguridad, climatización, iluminación, etc) y es necesario el acceso distribuido, por lo que CAN proporciona una buena implementación para la comunicación entre estos elementos.

Este protocolo está basado en el principio “productor/consumidor” donde cada equipo está siempre a la escucha y las transmisiones se realizan bajo el control de un equipo especial (el árbitro de bus). Las peticiones de información se construyen de acuerdo a una tabla de órdenes que contiene identificadores de variables. Al decodificar el nombre de variable asociado a la información que él produce, un dispositivo transmite los valores actuales correspondientes. Esta información es consumida por todos los receptores que reconocen el nombre de la variable. Este modo de funcionamiento garantiza que todos los dipositivos consumidores actualizan su información del proceso de forma simultánea. Todos los nodos, incluido el transmisor están activos mientras hay actividad en el bus, revisan si existen errores (hasta cinco diferentes chequeos de error) y fuerzan la retransmisión en caso de error. Todos los nodos deben aceptar el mensaje, en caso contrario se entiende que hay error. En el caso de los receptores, éstos envían un mensaje de “mensaje recibido” cuando el mensaje llega correctamente.

El bus CAN emplea un acceso al bus por prioridades mediante la técnica CSMA/CR (Carrier Sense Multiple Access / Collision Resolution), resolviendo los conflictos de acceso al bus mediante técnicas no destructivas, permitiendo un tiempo de inactividad garantizado en el caso de colisión. CAN no utiliza direcciones físicas para el nodo, dado que todos los nodos reciben todos los mensajes, cada uno de ellos decide si el mensaje va dirigido a él o no, esta decisión es tomada según la programación de cada nodo, o el hardware asociado. Pueden emplearse diferentes técnicas de gestión del bus como maestro/esclavo, multiplexado por división del tiempo (TDMA), o daysi chain.

Como caracterisitica esencial del bus CAN está la necesidad de uso de un protocolo para capas más elevadas capaz de realizar la conexión de la aplicación. CAN constituye únicamente una especificación de bajo nivel. Las posibilidades de CAN vienen determinadas en gran medida por el protocolo de las capas superiores. Este protocolo se elegirá dependiendo del mercado al que se oriente la aplicación, los requerimientos de tiempo real, etc. Por ejemplo, protocolos basados en CAN son CANopen y Devicenet.

Figura 3.5-3 *Formato de trama CAN (DLC = Data Length Code)*

Protocolo CANOPEN

Como se ha comentado, CAN necesita un protocolo de nivel superior para enlazar con las aplicaciones, este protocolo puede ser definido por cada usuario, o bien emplear algunos protocolos orientados a ciertas aplicaciones como CANopen, destinado para sistemas de control industrial. CANopen facilita el acceso a redes CAN dado que simplifica su empleo puesto que no es necesario controlar detalles tales como la temporización, control a nivel de bits, etc. Existen diferentes objetos orientados para datos en tiempo real (Process Data Objects), datos de configuración (Service Data Objects) y funciones especiales (mensajes de emergencia, de sincronismo) y datos de gestión de la red (arranque, control de errores, etc). Estas especificaciones incluyen diferentes perfiles predefinidos para dispositivos y entornos para aplicaciones indsutriales específicas, actualmente hay perfiles para módulos de entrada/salida genéricos, controladores de motores, medidas en dispositivos y control en lazo cerrado, encoders y válvulas hidráulicas. Las redes CANopen se han empleado en vehículos (comerciales, industriales, marítimos), equipos médicos y ferroviarios. La capa de aplicación que incorpora es muy flexible, permitiendo el desarrollo de aplicaciones a medida, pero en cambio, al tratarse de una capa estándar, permite compatibilizar muchos sistemas hardware.

Las principales ventajas de CANopen son:

- Se trata de una especificación abierta, con el estándar europeo EN50325-4. La organización CiA (CAN in Automation) es la encargada de promover los estándares relacionados con el bus CAN.
- Permite interoperabilidad entre diferentes dispositivos.
- Dispone de capacidad en tiempo real.
- Es un sistema modular que engloba dispositivos sencillos y complejos.
- Existen numerosas herramientas de programación y verificación.

Las caracterísiticas resumidas de CANopen son:

- La red es autoconfigurable.
- Los parámetros del dispositivo son fácilmente accesibles.
- Existe una sincronización entre dispositivos.

- La transferencia de datos es cíclica, activada por eventos y síncrona (tanto para lectura como escritura). [12]

Capítulo 4

Arquitectura y programación del proyecto

4.1 Arquitectura del Proyecto

Una de las características fundamentales que se perseguía durante la realización de este proyecto era permitir una gran flexibilidad en el diseño de la fuente ornamental. Flexibilidad en el número de variadores, número de luces y bombas, y flexibilidad también en las aplicaciones y servicios. El diseño hardware ha sido condicionado por esta flexibilidad, manteniendo todas las prestaciones que ofrecía la antigua configuración.

Inicialmente se trabajó sobre una arquitectura en la que el PLC estaba embebido en la pantalla, con el terminal gráfico, XBT-GC2230.

Esta configuración permitía añadir hasta siete tarjetas de entradas/salidas, modulos de comunicaciones, etc. Además de las 16 entradas digitales y 16 salidas digitales que tiene integradas.

Las primeras pruebas se hicieron con este terminal, más un módulo de entradas digitales, ya que con las 16 de serie, no era suficiente. Con los variadores la comunicación se hacía a través del puerto serie y en protocolo Modbus.

Este protocolo no daba la flexibilidad que se buscaba. El número de variadores no podía ser variable y tener a la vez una comunicación estable. Para conseguir la flexibilidad deseada en cuanto al número de variadores, éstos comunicarán con el PLC

vía CanOpen, que gracias a ser una red autoconfigurable permite cambiar el número de variadores sin modificar el código.

Se estudió la posibilidad de añadir a la pantalla la tarjeta de comunicaciones CANopen, pero ésta impide añadir otros módulos, por lo que el número de entradas digitales no era suficiente.

Por otro lado la pantalla con PLC daba algunos problemas a medida que el código se hacía más complejo.

Por todos estos motivos, se estudiaron otras opciones de un coste similar.

Finalmente, la arquitectura sobre la que se trabaja en este documento, se trata de un autómata M238, una pantalla HMISTU855, y en comunicación con hasta 8 variadores ATV312 mediante CANopen. Se puede ver la arquitectura de este proyecto en el siguiente esquema.

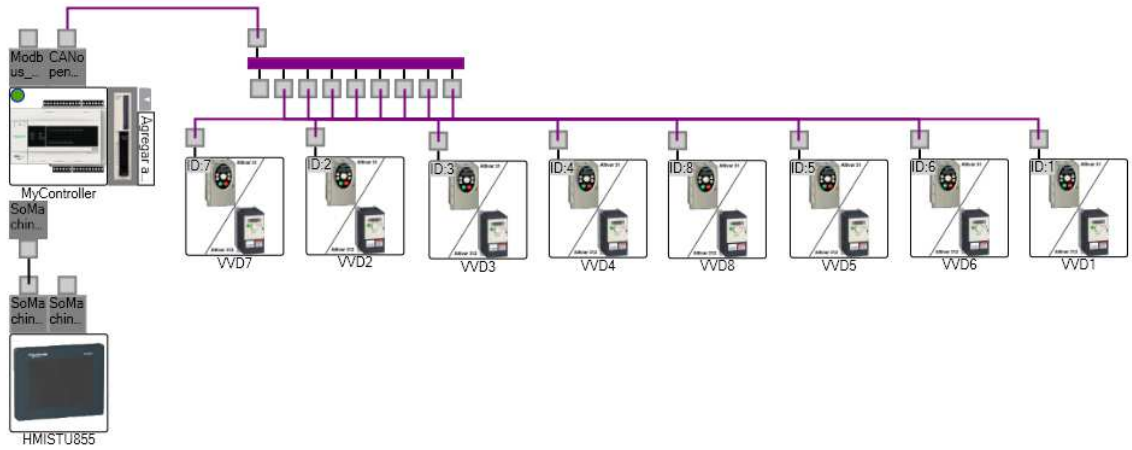


Figura 4.1-1 *Arquitectura hardware del proyecto*

4.1.1 **ATV312**

El variador compacto ALTIVAR 312 ofrece potencias de hasta 15 kW, con lo que cubriría perfectamente las necesidades del proyecto. Para cada posible fuente habrá que hacer el cálculo que defina exactamente que variador habría que seleccionar, pero esto no afecta a la programación de los mismos ni al proyecto en cuestión.

Figura 4.1-2 *Variador de velocidad Altivar 312 de Schneider Electric*

Este variador tiene como principales cualidades la calidad y robustez, para trabajar en cualquier tipo de entorno, y aporta grandes avances en comunicaciones de diálogo. Siempre trae incluidas de base Modbus y CanOpen.

Para un cableado más sencillo de las comunicaciones existe la opción de instalarle una tarjeta daysi chain. En las pruebas realizadas en el laboratorio así se hizo. Con estas tarjetas no habría necesidad de switch. Con un cable CanOpen se une el PLC con la tarjeta que se instala dentro del variador y otro sale hasta el siguiente variador, así hasta el último en el que se pone una resistencia de 120 ohmios para cerrar el bus.

Figura 4.1-3 *Variadores ATV312 con tarjeta de conexionado Daysi Chain*

En el anexo 8.1

Manual para el usuario desarrollado en este proyecto, se explica en detalle toda la información necesaria para la puesta en marcha de los variadores, tanto de su cableado (potencia y comunicaciones), cómo de la programación inicial.

4.1.2 **HMISTU 855 y VijeoDesigner**

La familia de terminales de diálogo Magelis de Schneirder Electric, va asociada al software Vijeo Designer.

Los terminales Magelis son utilizados para el control, manejo, diagnóstico y ajuste de los datos de los PLC, que controlan equipos como sistemas de control, sistemas de identificación, variadores de velocida, etc.

Todos los terminales Magelis están provistos con un IP65, grado de protección necesario para un proyecto de estas caraterísticas.

Dentro de toda la gama de posibles terminales, para este proyecto se ha seleccionado la HMI STU 855, un terminal gráfico pequeño y táctil especialmente diseñado para aplicaciones sencillas dónde se prima el pequeño tamaño cómo es este caso, para reducir costes.

Cómo se puede ver en la Figura 4.1-4 *Magelis STU855 de Schneider Electric* el montaje es muy sencillo, no es necesario realizar cajeado en el panel dónde vaya a instalarse, con un pequeño círculo por dónde meter el bulón, y así engancharlo con la parte trasera valdría.

Esta pantalla comunica vía RS485 y Ethernet con diferentes protocolos de varios fabricantes.

Cómo se ha visto en el esquema de la figura anterior (Figura 4.1-1 **Arquitectura hardware del proyecto**), en este caso se comunica por el puerto serie (COM1) mediante el protocolo SoMachine con el PLC.

Las funciones principales de estos terminales son:

- Visualizar datos procedentes del automatismo,
- Modificar parámetros del automatismo,
- Dirigir el automatismo mediante mandos Todo o Nada.

Figura 4.1-4 *Magelis STU855 de Schneider Electric*

Vijeo Designer es una aplicación de software de última generación con la que el usuario puede crear paneles de operador y configurar parámetros operativos para dispositivos de la interfaz usuario-máquina (HMI). Este programa proporciona todas las herramientas necesarias para el diseño de un proyecto HMI desde la adquisición de datos hasta la creación y la visualización de sinopsis animadas.

Algunas de las caracterísitcas de supervisión de control que ofrece esta combinación son:

- Administrador visual de Alarmas.
- Tendencias con herramientas de análisis integradas.
- Capacidad de reporte.
- Modificaciones online de los parámetros de control.
- Mecanismo de procesamientos de recetas.

En el anexo 8.1

Manual para el usuario desarrollado en este proyecto, se explica en detalle toda la información necesaria para la puesta en marcha de las pantallas, tanto de su cableado (potencia y comunicaciones), cómo de todo lo necesario para su programación, además de todas sus caracterísitcas técnicas.

4.1.3 **M238 y SoMachine**

El PLC seleccionado finalmente para este proyecto es el M238, que al igual que la pantalla XBTGC de la configuración de la que se partió, pertenece a la nueva plataforma de Schneider Electric, especialmente diseñada para fabricantes de maquinaria.

Dentro de la oferta de Schneider Electric, este contrlador, queda en la parte central, en cuanto a prestaciones, cómo se puede ver en la Figura 4.1-5 *Oferta de control Schneider Electric*.

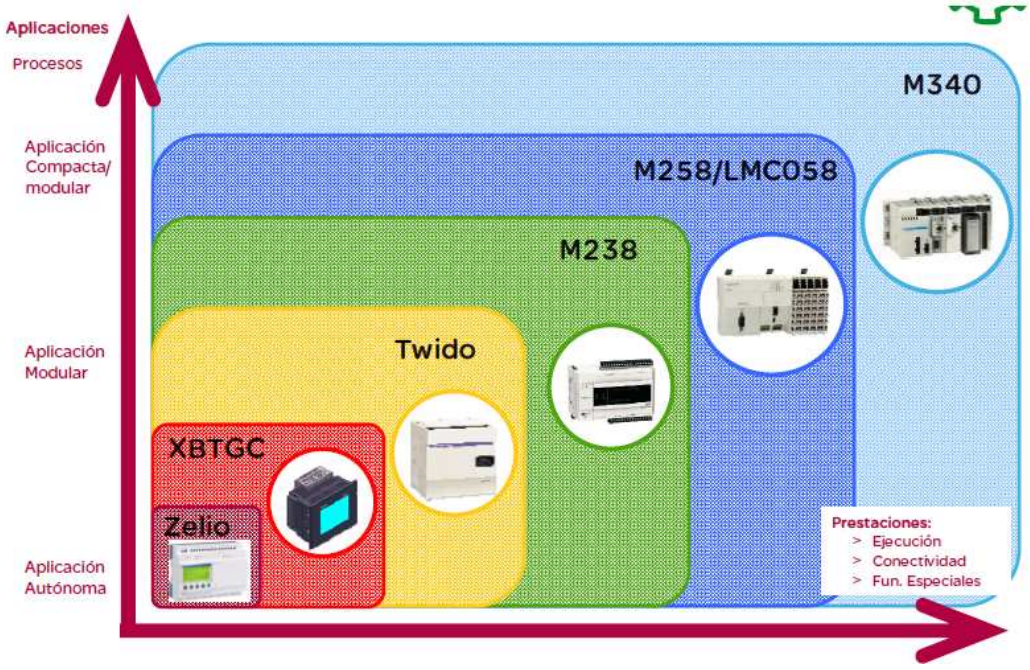


Figura 4.1-5 *Oferta de control Schneider Electric*

Los controladores lógicos compactos Modicon M238 ofrecen una solución muy completa con un tamaño compacto. Proporciona una gran flexibilidad respecto a su cableado (tornillos extraíbles, terminales de resorte y conectores HE10).

Entre sus características técnicas, cabe destacar la capacidad de comunicar con hasta 16 esclavos CANopen, ampliable con hasta 7 módulos de entradas/salidas, hasta 1024 kB de programa, etc.

Concretamente en las pruebas de laboratorio, se ha utilizado el PLC M238LFDC24DT: 24V CC, 6 entradas normales, 8 entradas rápidas, 4 salidas rápidas (100 kHz), 6 salidas de transistor (0,5 A) y con CANopen incrustado.

Figura 4.1-6 *Controlador lógico Modicon M238 de Schneider Electric*

El software de programación para esta nueva plataforma se llama como la misma, SoMachine, está basado en el lenguaje Codesys y se ajusta a la norma IEC 61131-3.

Este software ofrece en un sólo entono la configuración del PLC y de HMI integrada, quedando todo el proyecto en un solo archivo, y con variables compartidas entre el PLC y la HMI de un modo muy sencillo. Incluye también un simulador integrado para el PLC y HMI.

Las funciones de control, se pueden programar en seis lenguajes diferentes que cumple la normativa IEC 61131-3 (Lista de instrucciones IL, diagrama de contactos LD, diagrama de bloques de funciones FBD, sequential function chart / Grafcet SFC, lenguaje CFC Continuosu Function Chart y texto estructurado ST.

En el anexo 8.1

Manual para el usuario desarrollado en este proyecto, se explica en detalle toda la información necesaria para la puesta en marcha del PLC, tanto de su cableado (potencia y comunicaciones), cómo de todo lo necesario para su programación, además de todas sus caracterísitcas técnicas.

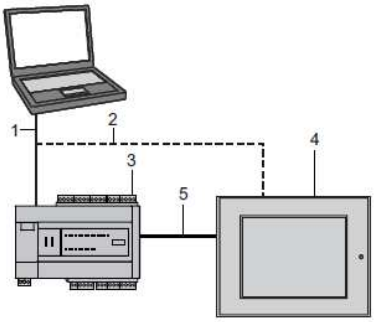
4.1.4 **Comunicaciones**

Casi todo lo referente a las comunicaciones ha sido explicado en apartados anteriores, tanto la descripción teórica en el capítulo 3.5-Comunicaciones, cómo las comunicaciones que se han utilizado y el porqué, visto en este mismo capítulo. Queda por mencionar las características técnicas, los parámetros de configuración de las comunicaciones del proyecto.

En el caso de la comunicación entre el PLC y la pantalla, estuvo claro en todo momento que la comunicación tenía que ser por el puerto serie y con el protocolo SoMachine. Éste es el preferido porque ofrece un acceso transparente al controlador de la máquina y HMI.

El protocolo SoMachine se utiliza para cualquier intercambio de datos entre el software de SoMachine (el PC) y los sitemas de tiempo de ejecución (controlador, HMI), y entre el controlador y HMI.

Las ventajas que ofrece directamente el uso de este protocolo son entre otras el uso de un sólo cable que permite acceder a cualquier HMI o controlador que forme parte de su máquina mediante la conexión de su PC de SoMachine a un solo dispositivo de la máquina. La conexión se direccionará a través de los diferentes dispositivos conectados. Esta conexión única con la máquina proporciona una ganancia en simplicidad al transferir los datos utilizando el mismo cable del PC a la máquina como se puede ver en la figura siguiente.



1 Conexión entre el PC de SoMachine y el controlador
2 Conexión alternativa entre el PC de SoMachine y HMI
3 Logic controller
4 HMI
5 Conexión de línea serie entre el controlador y HMI

Figura 4.1-7 *Conexionado Protocolo Transparente SoMachine*

Por otro lado el protocolo transparente SoMachine le permite definir las variables una sola vez en el proyecto y ponerlas a disposición de cualquier otro HMI o controlador mediante un mecanismo de publicación y suscriptor basado en nombres simbólicos. Una vez que las variables se han publicado, los demás HMI o controladores pueden suscribirse a ellas sin necesidad de volver a especificar la definición de las variables.

Es fundamental a la hora de comunicar diferentes dispositivos, que todos tengan los mismos parámetros de comunicación (velocidad de transmisión, paridad, número de bits de datos y bit de parada).

La configuración utilizada finalmente ha sido, la que se muestra en la figura, pero esta se puede modificar, siempre y cuando se haga del mismo modo en todos los dispositivos de la red.

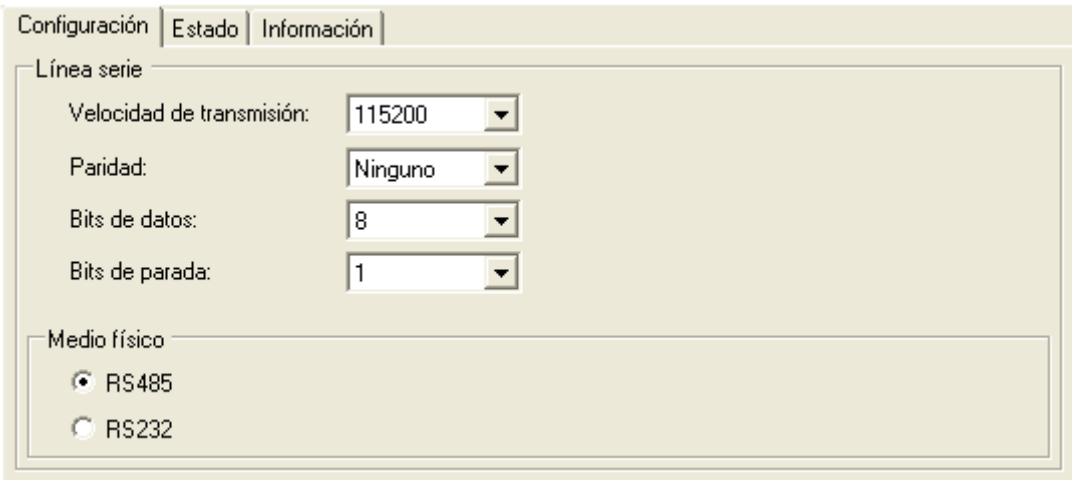


Figura 4.1-8 *Parámetros de comunicación PLC-HMI*

La última gran ventaja que ofrece el protocolo SoMachine es el acceso transparente a los dispositivos del bus de campo. La conexión única entre el PC y el controlador, da acceso a cada dispositivo conectado en CANopen. Desde la interfaz de usuario única de SoMachine, es posible configurar dispositivos remotos sin conexión y sintonizarlos en línea.

En este caso serán los variadores de velocidad ATV312, los dispositivos conectados en CANopen.

Cómo ya se ha visto, CANopen es un estándar internacional que define un protocolo de comunicación industrial para conectarse con equipos industriales vía red CAN. El conexionado se hará a través del CANbus, cable de par trenzado apantallado con una resistencia de 120 ohm, señales CAN_H, CAN_L, longitud máxima del bus de 1000m y la velocidad de transmisión depende de la longitud del bus, todos los esclavos tienen que estar configurados con la misma velocidad y no puede haber dos equipos con el mismo número de esclavo.

4.2 Programación del proyecto: SoMachine

La navegación dentro del Software SoMachine es intutitiva y muy visual. El interface de usuario está optimizado para que en los diferentes pasos del proyecto, se habiliten las herramientas necesarias para ese paso. El interfaz de usuario habilitará las opciones que se puedan realizar en cada paso.

Con la ayuda de las pestañas de navegación se puede seleccionar en que paso se está en cada momento de desarrollo del proyecto.

Figura 4.2-1 *Navegación por pesatñas del SoMachine*

En la pestaña ‘Propiedades’ se pueden rellenar diferentes campos con información específica de cada proyecto.

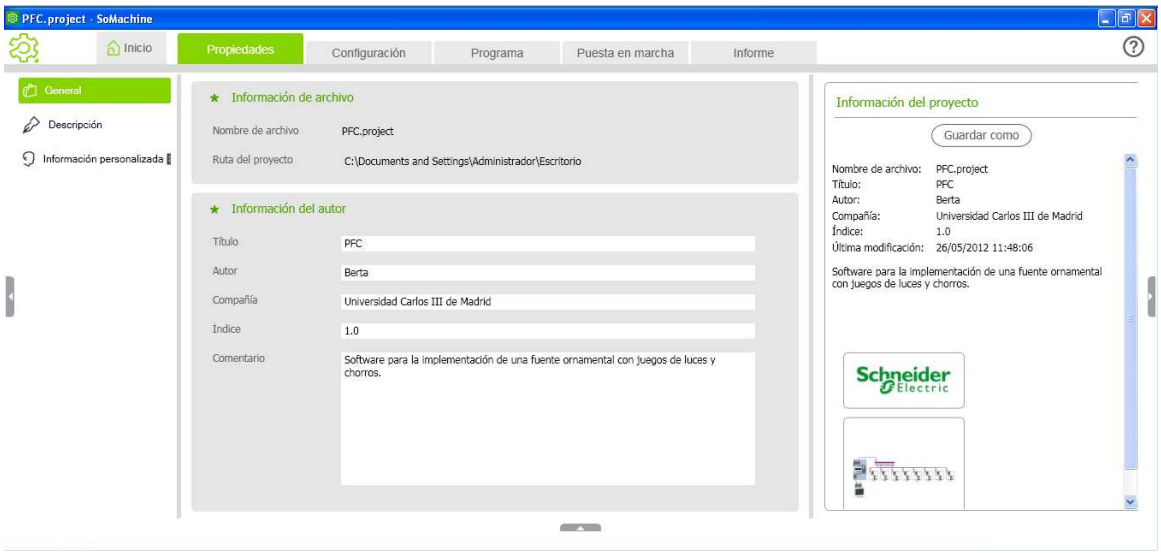


Figura 4.2-2 *Pestaña Propiedades*

Después de describir el proyecto, en la pestaña ‘Configuración’ se tiene que crear la configuración de la arquitectura que va a tener la máquina (como mínimo el controlador que se va a utilizar). Gracias a que el proceso de configuración es muy gráfico, se hace muy fácil incluir módulos de ampliación, buses de comunicaciones o HMI.

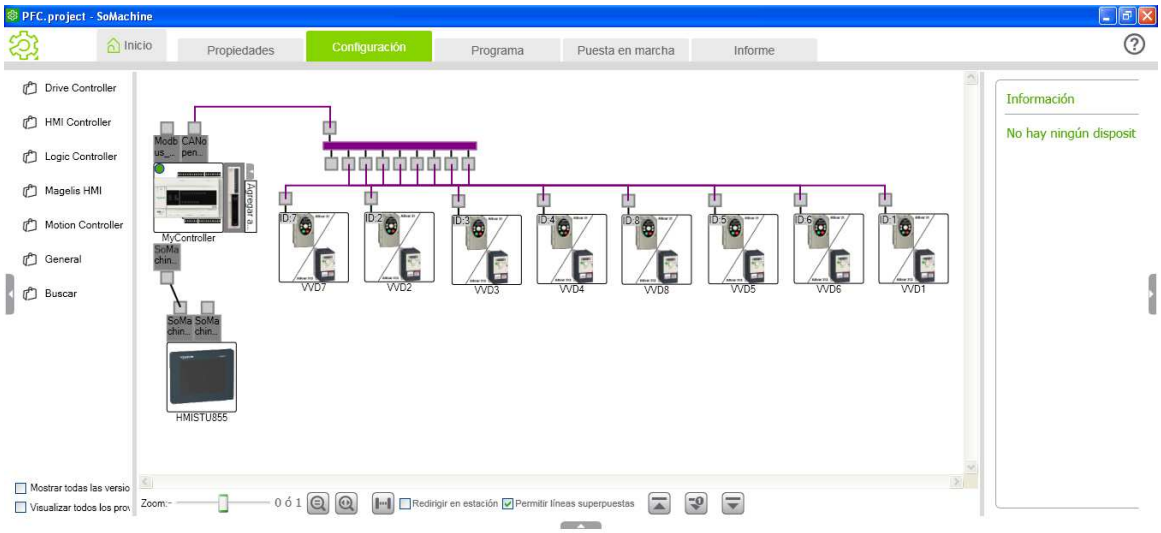


Figura 4.2-3 *Pestaña Configuración*

La pestaña de ‘Programa’ abre el proyecto y permite realizar la programación de este. Todas las funcionalidades de CoDeSys están disponibles en esta ventana.

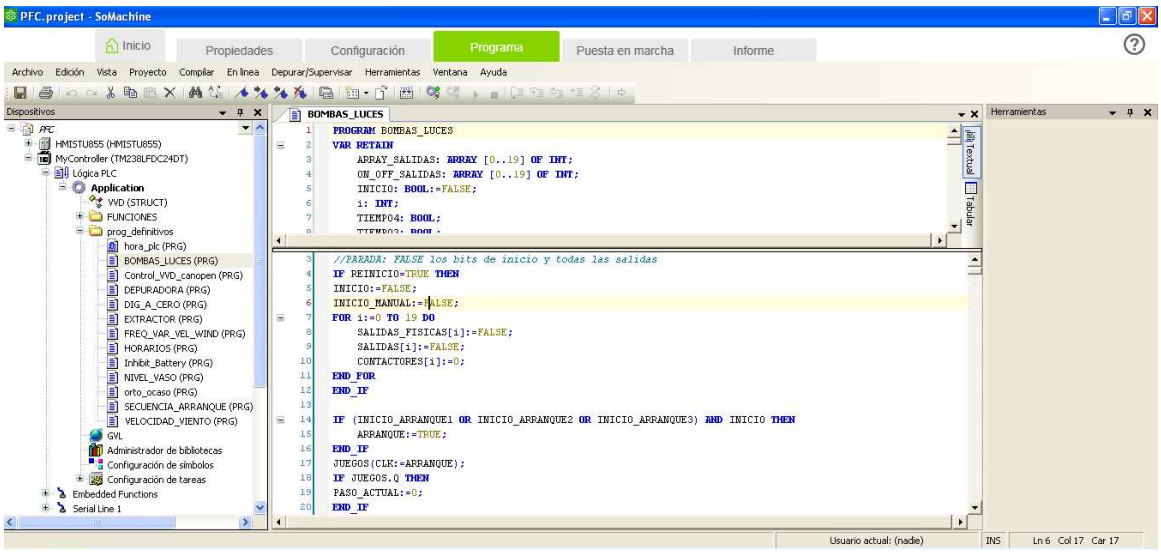


Figura 4.2-4 *Pestaña Programa*

Con la pestaña 'Puesta en Marcha' se puede tener acceso online al controlador y se puede monitorizar el estatus de todos los equipos conectados y permite descargar la aplicación o el firmware.

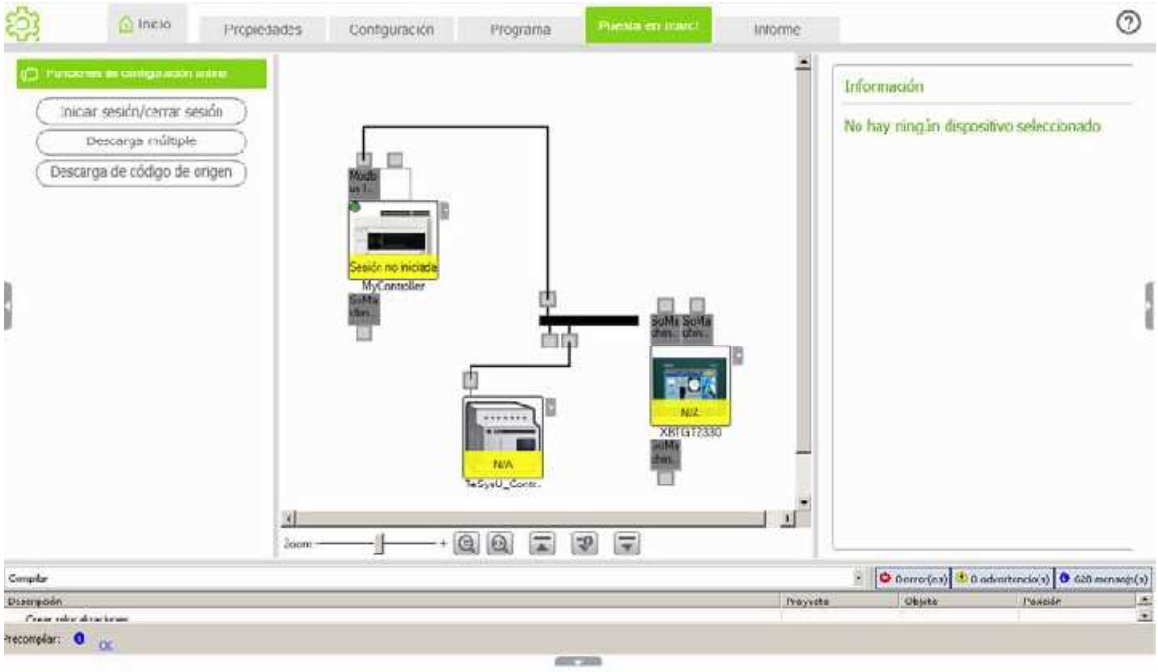


Figura 4.2-5 *Pestaña Puesta en Marcha*

Toda la programación que se ha realizado se puede ver en el programa anexado a este proyecto. En este apartado se verá resumidamente la estructura de la aplicación y más en detalle la programación de las subrutinas desarrolladas para hacer funcionar las fuentes ornamentales según lo especificado.

4.2.1 **Estructura de la Aplicación.**

Cómo se puede ver en esta figura, la aplicación se divide en varios apartados. Además de los programas y funciones creados (de ahora en adelante llamados POU, Program Organization Unit), que se agrupan en carpetas, están las estructuras de datos creadas por el usuario, la lista de variables globales (GVL), el administrador de bibliotecas, el configurador de simbolos y la configuración de tareas, de dónde cuelgan todas las tareas creadas, en este caso sólo la MAST.

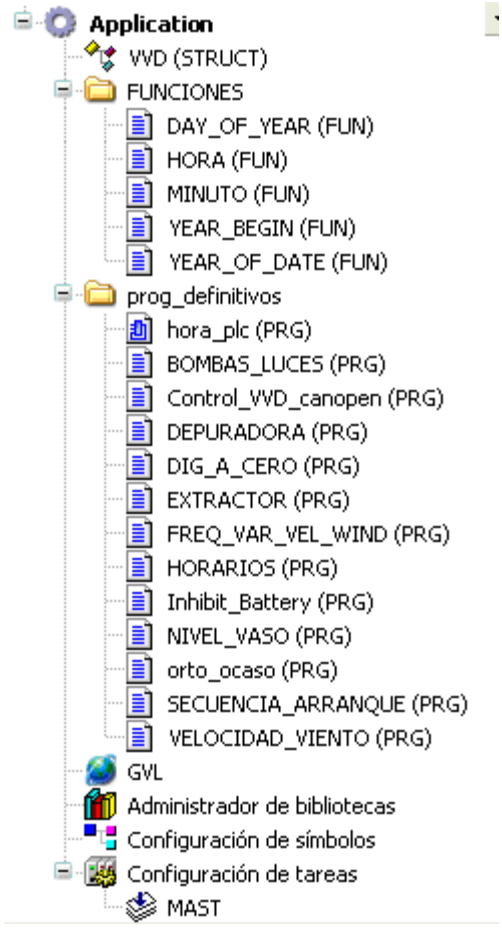


Figura 4.2-6 *Lógica programada del PLC*

Para este proyecto se ha creado una estructura de datos (**DUT, Data Unit Type**) llamada VVD. Una estructura de datos definida por el usuario, es una agrupación de variables de tipos de datos diferentes o de variables de tipos de datos iguales.

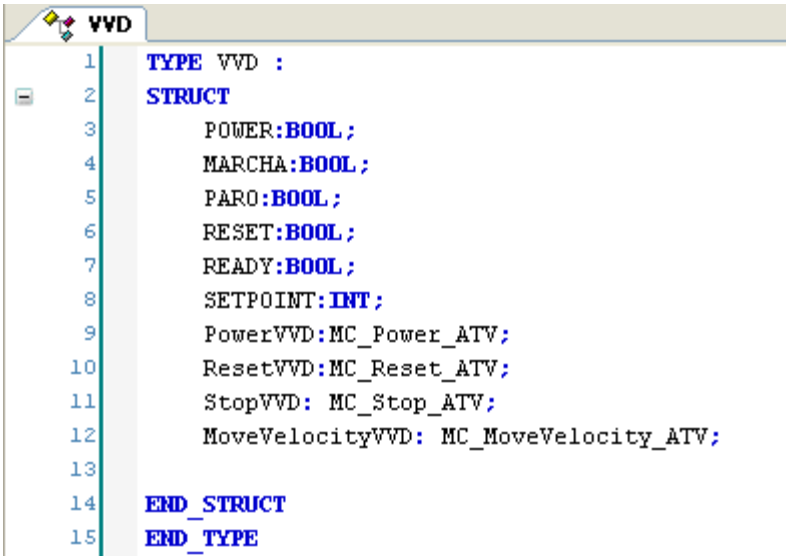


Figura 4.2-7 *Estructura de datos para los Variadores de Velocidad*

Esta es la estructura que ha sido creada para el manejo y la parametrización de los variadores. Los datos de tipo MC_, son propios del SoMachine, implementados para el manejo de los variadores ATV de un modo muy sencillo.

Una vez añadida la estructura de datos en el programa se pueden crear variables con ese tipo de dato. En concreto esta DUT se ha utilizado en la rutina de control de los variadores *Control_VVD_canopen*, creando una matriz de 8 datos de tipo VVD, a la que se ha llamado ATV.

ATV:ARRAY[0..8] OF VVD;

Se puede ver a continuación fragmentos de la rutina dónde se lleva a cabo todo el control de los variadores, de un modo sencillo gracias a esta estructura de datos.

```
//SECUENCIA DE ARRANQUE PROGRESIVO:
IF REINICIO THEN
   PARO_VARIADORES:=TRUE;
   MARCHA_VARIADORES:=FALSE;
END_IF

IF ATV_ARRANQUE[0]=TRUE THEN
   ATV[0].MARCHA:=TRUE;
END_IF
IF ATV_ARRANQUE[1]=TRUE THEN
   ATV[1].MARCHA:=TRUE;
END_IF
…
IF ATV_ARRANQUE[8]=TRUE THEN
   ATV[8].MARCHA:=TRUE;
```

```
    END_IF

    IF (FTE_ON=FALSE AND MANUAL=FALSE) OR PARO_VIENTO=TRUE
THEN
        PARO_VARIADORES:=TRUE;
        MARCHA_VARIADORES:=FALSE;
    ELSIF ARRANQUE THEN
     PARO_VARIADORES:=FALSE;
    END_IF

    //power
        ATV[0].PowerVVD(Enable:= atv[0].POWER, Axis:= VVD1 , Status=> ,
Error=> );
        ATV[1].PowerVVD(Enable:= atv[1].POWER, Axis:= VVD2 , Status=> ,
Error=> );
        …
        ATV[7].PowerVVD(Enable:= atv[7].POWER, Axis:= VVD8 , Status=> ,
Error=> );

    //RESET
    ATV[0].ResetVVD(Execute:= atv[0].reset,Axis:= VVD1,Done=> ,Busy=> ,Error=>
);
    ATV[1].ResetVVD(Execute:= atv[1].reset,Axis:= VVD2,Done=> ,Busy=> ,Error=>
);
    …
    ATV[7].ResetVVD(Execute:= atv[7].reset,Axis:= VVD8,Done=> ,Busy=> ,Error=>
);

    //Paro
    ATV[0].StopVVD(Execute:=ATV[0].PARO, Axis:= VVD1);
    ATV[1].StopVVD(Execute:=ATV[1].PARO, Axis:= VVD2);
    …
    ATV[7].StopVVD(Execute:=ATV[7].PARO, Axis:= VVD8);

    IF reset_total THEN
        atv[0].reset:=TRUE;
        atv[1].reset:=TRUE;
```

```
…
    atv[7].reset:=TRUE;
ELSE
    atv[0].reset:=FALSE;
    atv[1].reset:=FALSE;
…
    atv[7].reset:=FALSE;
END_IF


CASE numero_variadores OF
(* Esta instrucción CASE que se utiliza a lo largo del programa es necesaria para ofrecer la flexibilidad requerida en cuanto al número de variadores que podrá ser desde 0 hasta 8*)
1:ATV[0].POWER:=TRUE;
2:ATV[0].POWER:=TRUE; ATV[1].POWER:=TRUE;
…
8:ATV[0].POWER:=TRUE; ATV[1].POWER:=TRUE; ATV[2].POWER:=TRUE; ATV[3].POWER:=TRUE; ATV[4].POWER:=TRUE; ATV[5].POWER:=TRUE; ATV[6].POWER:=TRUE; ATV[7].POWER:=TRUE;
END_CASE


//MARCHA VARIADORES
//VVD1
ATV[0].MoveVelocityVVD (Execute:= ATV[0].MARCHA,Velocity:= ATV[0].SETPOINT, Axis:= VVD1);
ATV_SETPOINT[0]:=FREQ_VAR[0];
IF ATV[0].MoveVelocityVVD.InVelocity AND ATV[0].MARCHA THEN
   ATV[0].MARCHA:=FALSE;
END_IF
IF ATV[0].StopVVD.Done AND ATV[0].PARO THEN
   ATV[0].PARO:=FALSE;
END_IF
…
//VVD8
ATV[7].MoveVelocityVVD (Execute:= ATV[7].MARCHA,Velocity:= ATV[7].SETPOINT, Axis:= VVD8);
ATV_SETPOINT[7]:=FREQ_VAR[7];
```

```
    IF ATV[7].MoveVelocityVVD.InVelocity AND ATV[7].MARCHA THEN
        ATV[7].MARCHA:=FALSE;
    END_IF
    IF ATV[7].StopVVD.Done AND ATV[7].PARO THEN
        ATV[7].PARO:=FALSE;
    END_IF

    //INSTRUCCIÓN DE PARO
    IF PARO_VARIADORES THEN
        CASE numero_variadores OF
            1: ATV[0].PARO:=TRUE;
            2: ATV[0].PARO:=TRUE;ATV[1].PARO:=TRUE;
            …
            8:
ATV[0].PARO:=TRUE;ATV[1].PARO:=TRUE;ATV[2].PARO:=TRUE;ATV[3].PARO
:=TRUE;ATV[4].PARO:=TRUE;ATV[5].PARO:=TRUE;ATV[6].PARO:=TRUE;ATV[
7].PARO:=TRUE;
        END_CASE
    ELSE
    ATV[0].PARO:=FALSE; ATV[1].PARO:=FALSE; ATV[2].PARO:=FALSE;
ATV[3].PARO:=FALSE; ATV[4].PARO:=FALSE; ATV[5].PARO:=FALSE;
ATV[6].PARO:=FALSE; ATV[7].PARO:=FALSE;
    END_IF

    //INSTRUCCIÓN DE MARCHA
    IF MARCHA_VARIADORES  AND (FTE_ON OR HORARIOS_OFF) THEN
        CASE numero_variadores OF
            1: ATV[0].MARCHA:=TRUE;
            2: ATV[0].MARCHA:=TRUE;ATV[1].MARCHA:=TRUE;
            …
            8: ATV[0].MARCHA:=TRUE; ATV[1].MARCHA:=TRUE;
ATV[2].MARCHA:=TRUE; ATV[3].MARCHA:=TRUE; ATV[4].MARCHA:=TRUE;
ATV[5].MARCHA:=TRUE; ATV[6].MARCHA:=TRUE; ATV[7].MARCHA:=TRUE;
        END_CASE
    //  REPOSO:=FALSE;
    ELSE
```

```
    ATV[0].MARCHA:=FALSE; ATV[1].MARCHA:=FALSE;
ATV[2].MARCHA:=FALSE; ATV[3].MARCHA:=FALSE;
ATV[4].MARCHA:=FALSE; ATV[5].MARCHA:=FALSE;
ATV[6].MARCHA:=FALSE; ATV[7].MARCHA:=FALSE;
    END_IF

    //FRECUENCIA PARA LA PUESTA EN MARCHA:
    FOR c:=0 TO (numero_variadores-1) DO
       IF SALIDAS_AUT_MAN[c+20] THEN
             FREQ_MAX[c]:=FREQ_PASO0[c];
       END_IF
    END_FOR
```

En este caso el array ATV se trata de una variable local, es decir, que sólo se utiliza en esa unidad de programa o subrutina.

Otro lugar dónde declarar las variables es la **GVL o lista de variables globales**, las variables que se declaren en esta lista se pueden utilizar en toda la aplicación. Se pueden crear un máximo de tres listas de variables globales por aplicación, para este proyecto se ha creado solamente una.

```
VAR_GLOBAL RETAIN

//DESACTIVACIÓN MANUAL DE DEPURADORA, EXTRACTOR, RELOJES, ANEMÓMETRO
DEPURADORA_OFF:BOOL:=0;
EXTRACTOR_OFF:BOOL:=0;
HORARIOS_OFF:BOOL:=0;
ANEMOMETRO_OFF:BOOL:=0;

ASIGNACION_DEP:BOOL:=0;
ASIGNACION_EXTRACT:BOOL:=0;
DEPURADORA_ON:BOOL:=0;
EXTRACTOR_ON:BOOL:=0;


    //canopen

    ATV_SETPOINT:ARRAY[0..8] OF INT;
    ATV_OUTSPEED:ARRAY[0..8] OF INT;
    ON_OFF_SALIDAS_2: ARRAY [0..19] OF INT;
    ATV_ARRANQUE: ARRAY [0..8] OF BOOL;
    INICIO_MANUAL:BOOL;
    INICIO_ARRANQUE1:BOOL;
    INICIO_ARRANQUE2:BOOL;
    INICIO_ARRANQUE3:BOOL;
    JUEGOS:F_TRIG;
END_VAR
```

Figura 4.2-8 *Fragmento de la GVL: Global Variable List*

Las variables que aparezcan en esta lista serán de tipo RETAIN, porque así se ha definido.

Las variables Retain mantienen su valor tras una finalización incontrolada o tras un comando en línea ' Reset en caliente ', así como tras una desconexión y conexión (Reboot) normal del control. Ante un reinicio del programa se continúa trabajando con los valores memorizados. Todas las otras variables en este caso se reinicializan, bien sea con sus propios valores inicializados, o con las inicializaciones predeterminadas.

El '**Administrador de bibliotecas**' es un objeto que aparece automáticamente creado en el árbol de la aplicación.

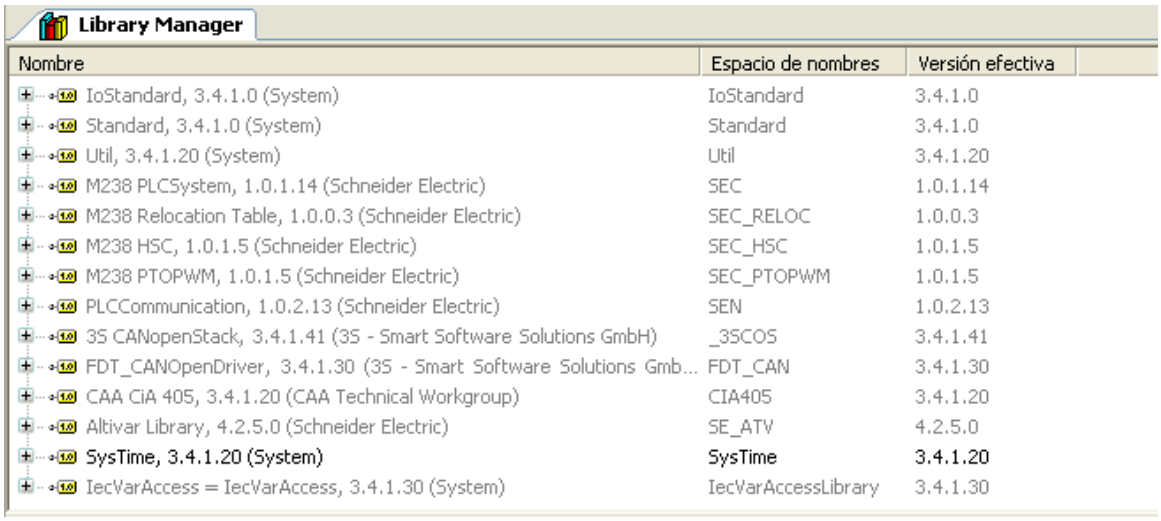


Figura 4.2-9 *Administrador de Librerías del proyecto*

El administrador de bibliotecas indica las librerías que se han incluido en la aplicación. Las que se agregan automáticamente y las que agrega el usuario. Para el proyecto sólo se ha añadido la librería SysTime, además de las que se agregan automáticamente. Por ejemplo, al incluir un variador ATV en la arquitectura hardware, automáticamente se añade la librería *Altivar Library* incluyendo las funciones y los tipos de dato MC para el manejo de variadores.

Las funciones de Systime se utilizan para obtener y ajustar el reloj de tiempo real. Contiene además un tipo de dato que es el que se ha utilizado a la hora de introducir y mostrar por pantalla fecha y hora de un modo muy sencillo. El dato SYSTIMDATE:

TYPE **SYSTIMEDATE**

Description:
Time and date in structured

Parameters:

Name	Type	Comment
wYear	VAR	Year (e.g. 2006)
wMonth	VAR	Month (1..12: January = 1, December = 12)
wDay	VAR	Day of month (1..31)
wHour	VAR	Hours after midnight (0..23)
wMinute	VAR	Minutes after hour (0..59)
wSecond	VAR	Seconds after minute (0..59)
wMilliseconds	VAR	Milliseconds after second (0..999). Optional!
wDayOfWeek	VAR	Day of week (1..7: Monday = 1, Sunday = 7
wYday	VAR	Day of year (1..365): January 1 = 1, December 31 = 364/365

Tabla 4.2-1 *Dato SYSTIMEDATE de la librería de SysTime de SoMachine*

Este tipo de dato se utiliza por ejemplo en el POU *hora_plc*, que es el único que se ha programado en un leguaje distinto de texto estructurado. Este POU fue programado en CFC (Continuous Function Chart), que es muy parecido al Bloque de Funciones, sólo que en este se pueden colocar los bloques libremente por el espacio de trabajo.

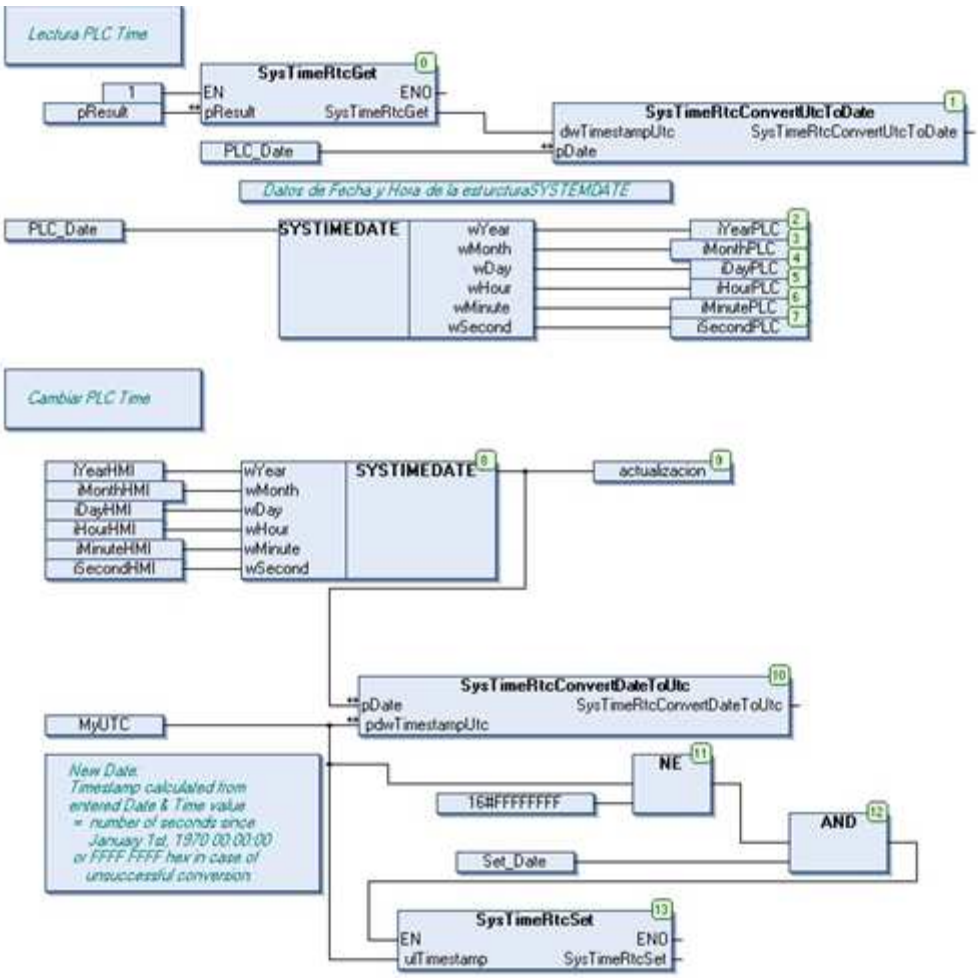


Figura 4.2-10 *POU hora_plc en CFC*

La '**configuración de símbolos**' se emplea, para crear símbolos con determinados derechos de acceso, con los que se puede acceder externamente a variables de proyecto (aplicaciones), en este caso desde la aplicación de la pantalla HMI. Una descripción de los símbolos se pone a disposición en un archivo xml (archivo de símbolo) del directorio del proyecto y simultáneamente se carga con la aplicación sobre el control.

Para ello se crea una "lista de símbolos", que por una parte se exporta a un archivo XML en el directorio del proyecto y por la otra, durante la descarga de la aplicación se carga en el sistema de destino en un archivo no visible para el usuario.

A esta lista solamente se añadirán las variables que van a ser utilizadas en la aplicación del VijeoDesigner.

En el objeto configuración de símbolos, aparecen dos ventanas, una dónde se pueden ver todas las variables existentes en nuestro proyecto de SoMachine, organizadas por POU´s o GVL, dónde hayan sido declaradas, y en la derecha las variables que se quieren compartir. Esto simplifica muchísimo la compartición de variables entre nuestro PLC y el HMI, además en ambos sitios aparecerán con el mismo nombre, y en el VijeoDesigner no será necesario crearlas otra vez.

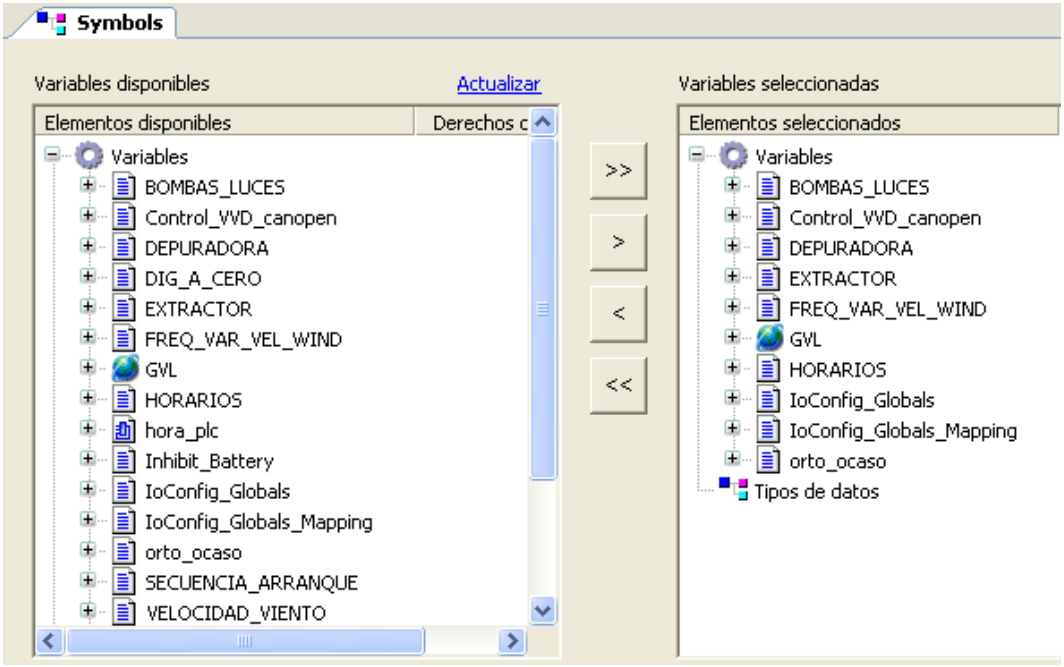


Figura 4.2-11 *Configuración de símbolos del proyecto*

Por último dentro de la lista de objetos de la aplicación aparece siempre la '**configuración de tareas**'. Ésta controla la ejecución de los programas en el proyecto. La tarea MAST, se crea por defecto por el sistema en el momento que se crea un proyecto nuevo.

Una tarea es un grupo de secciones de programa y subrutinas ejecutadas de una manera cíclica o periódica por la tarea principal MAST, y periódica por la tarea FAST (tarea cíclica con un tiempo de ejecución menor que la tarea MAST).

El 'Configurador de Tareas' permite definir una o varias tareas que se ejecutarán en el controlador. Las tareas controlan la ejecución de la aplicación.

Este proyecto sólo tiene una tarea MAST, en la que se ejecutarán todos los POUs en el orden que se ha considerado más adecuado. Esta tarea se ha definido con una prioridad 15 (esto es irrelevante habiendo sólo una tarea), y de tipo Ejecución libre, esta tarea cíclica se ejecuta al ponerse el PLC en estado de Run. No tiene un intervalo de ejecución definido por el usuario, pero internamente tendrá que estar entre T#: 1...1000 ms.

También se definie aquí el tiempo de Watchdog (100ms) y la sensibilidad de éste (1). La sensibilidad configura el número de veces (continuo) que el watchdog tiene que producirse antes de que salte el evento.

4.2.2 **POUs: Program Object Units.**

Un POU es una sección de programa dónde se escribirá el código de programa. Hay tres tipos de POU diferentes:

Tipos de POU	Descripción
Programa	Devuelve diferentes valores durante su procesado. Todos los valores se mantienen de un ciclo de máquina al anterior. Este puede ser llamado por otro POU.
Bloque de funciones	Devuelve diferentes valores durante su procesado de un programa. Es opuesto a la función, porque puede tener más de un valor de salida y necesita variables internas que tiene que ser persistente entre un ciclo de ejecución y el siguiente. Este tiene que ser llamado (una o más veces) desde un mismo POU programa.
Función	Devuelve una salida y no tiene variables internas persistentes, solo variables temporales entre un ciclo de ejecución y el siguiente no persisten los datos).

Tabla 4.2-2 *Tipos de POU*

Cada vez que se crea un POU, se define el nombre, el tipo y el lenguaje en el que se quiere implementar. En este proyecto la inmensa mayoría de ellos están programados en texto estructurado, por ser este lenguaje el más potente de los seis.

Para este proyecto se han utilizado los siguientes POU´s:

- **Funciones**

o YEAR BEGIN: devuelve la fecha del uno de enero del año introducido.

o YEAR OF DATE: devuelve el año de la fecha introducida.

o DAY OF YEAR: haciendo uso de las dos funciones anteriores, calcula el día del año de la fecha introducida.

o HORA: calcula la hora desde el parámetro tiempo.

o MINUTO: calcula el minuto desde el parámetro tiempo.

- **Programas**

o DIG A CERO: resetea la pantalla de selección de juegos cuando se solicita desde la pantalla.

o HORA PLC: implementado cómo se vio anteriormente en CFC (Continuous Function Chart).

o HORARIOS: Desde el HMI se introducen dos horarios de encendido y pagado, uno de lunes a viernes, y otro para sábado y domingo. Se configuran de manera separada, con lo que podrán tener franjas de horario diferentes las luces y el agua, teniendo en cuenta siempre que no pueden estar las luces encendidas si no lo están las bombas, para evitar un posible calentamiento de las luces incluso que lleguen a quemarse si no hay agua en el vaso.

o ORTO OCASO: Esta subrutina, esta basada en un algoritmo matemático de cálculo del orto y el ocaso. Calcula a partir de la fecha y las coordenadas en las que se localice la fuente (dato que también se introduce por pantalla en la configuración inicial de la fuente) a que hora sale y se pone el sol. De este modo, en la configuración de los horarios de las luces, se puede elegir que se enciendan con la puesta del sol y se apaguen cuando éste salga. Esta hora irá cambiando cada día automáticamente.

- VELOCIDAD VIENTO: programa muy sencillo que cálcula la velocidad del viento a partir de los flancos, señales digitales que se tienen cómo entrada del anemómetro.

- FREQ VAR VEL WIND: En la configuración de los variadores se tiene que tener en cuenta la velocidad del viento, cuanto mayor sea el viento, menor ha de ser la velocidad de los variadores, para evitar la perdida de agua que el viento puede producir empujando el agua fuera del vaso de la fuente.

Los variadores funcionarán a la velocidad que se les haya definido para cada paso, pero si el viento supera un determinado nivel, la frecuencia de los variadores se verá forzada a un valor menor. Así habrá hasta tres niveles de viento críticos, con frecuencias para los variadores asignadas. Si el viento supera el tercer nivel, los variadores directamente se pararan hasta que éste recupere un valor normal.

- NIVEL VASO: A partir de las entradas de los tres sensores de nivel que habrá en el vaso, se calcula si el vaso está lleno, medio lleno, medio vacío, o vacío (NIVEL_VASO= 3, 2, 1 o 0 respectivamente). En función de esta variable, se activará la entrada de agua cuando el vaso este vacío o medio vació, y si se llega a vacío, la fuente se parará.

Estos niveles se representarán gráficamente en la pantalla.

- DEPURADORA: La depuradora cómo una salida más tendrá configurable un horario de encendido y uno de apagado (se seleccionan los minutos que ha de estar encendida). Este POU se encarga de encender y apagar la depuradora cuándo sea necesario, gestionando la salida digital y enviando una señal que recogerá el HMI para representarlo también por pantalla.

- EXTRACTOR: Exactamente igual que el POU depuradora pero para contolar el extractor de la fuente.

- BOMBAS LUCES: Este POU es el más activo. Desde aquí se gestiona el encendido y apagado de todas las salidas, fuentes y contactores, y las frecuencias de los variadores para cada paso definido desde pantalla. También el arranque sin juegos (Arranque manual), y se lanza el arranque progresivo que llevará a cabo en la sequencia de arranque.

- CONTROL VVD CANOPEN: también visto en el apartado anterior, en este POU se lleva a cabo todo el control (Marcha, paro, control de velocidades…) de los variadores.

- SECUENCIA ARRANQUE: Una fuente ornamental no es capaz de arrancar de golpe todas las luces y bombas. Esto ha de hacerse de un modo progresivo que se encargará de gestionar este POU. Por pantalla de sefine el tiempo que se quiere esperar entre contactor y contactor y entre variador y variador. La secuencia de encendido a seguir será la siguiente: primero se encenderán todas las luces, a continuación empezarán los contactores uno por uno, con el intervalo de tiempo definido entre cada activación, y por último los variadores del mismo modo que los contactores.

En el Anexo 8.2 **Secuencias de código**, se encuentra el código implementado para las rutinas que se han considerado más significativas.

4.2.3 **Asignación de entradas salidas.**

En el software SoMachine, la asignación de las entradas y salidas a una variable del código es también muy sencilla.

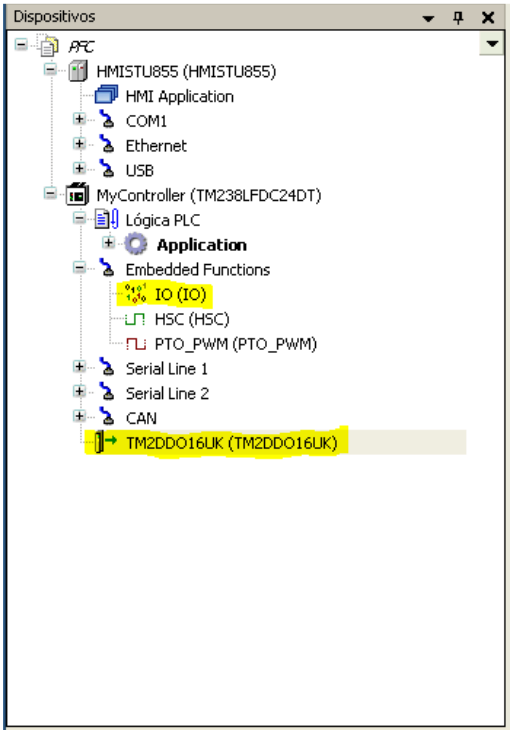
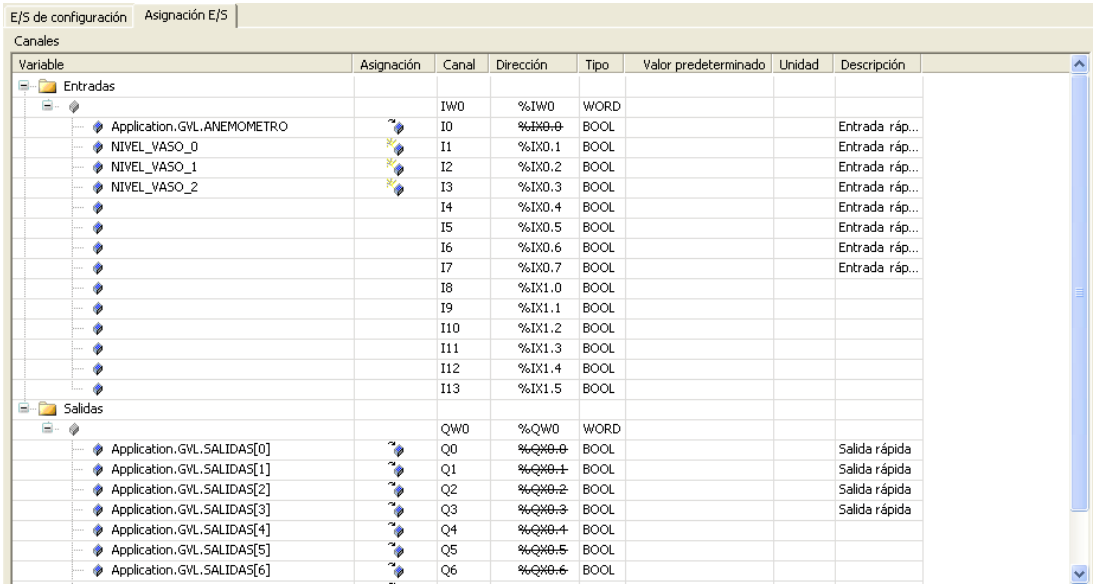


Figura 4.2-12 *Dispositivos de E/S*

Accediedo a los dispositivos de entradas y salidas, en la figura anterior están remarcadas las de este proyecto, tanto las entradas y salidas que vienen embebidas en este PLC (IO), cómo el módulo de expansión de salidas digitales (TM2DDO16UK).

La asignación se realiza simplemente escribiendo el nombre de la variable que se ha utilizado en el código al lado de cada variable de entrada o de salida.



Variable	Asignación	Canal	Dirección	Tipo	Valor predeterminado	Unidad	Descripción
Entradas							
		IW0	%IW0	WORD			
Application.GVL.ANEMOMETRO		I0	~~%IX0.0~~	BOOL			Entrada ráp...
NIVEL_VASO_0		I1	%IX0.1	BOOL			Entrada ráp...
NIVEL_VASO_1		I2	%IX0.2	BOOL			Entrada ráp...
NIVEL_VASO_2		I3	%IX0.3	BOOL			Entrada ráp...
		I4	%IX0.4	BOOL			Entrada ráp...
		I5	%IX0.5	BOOL			Entrada ráp...
		I6	%IX0.6	BOOL			Entrada ráp...
		I7	%IX0.7	BOOL			Entrada ráp...
		I8	%IX1.0	BOOL			
		I9	%IX1.1	BOOL			
		I10	%IX1.2	BOOL			
		I11	%IX1.3	BOOL			
		I12	%IX1.4	BOOL			
		I13	%IX1.5	BOOL			
Salidas							
		QW0	%QW0	WORD			
Application.GVL.SALIDAS[0]		Q0	~~%QX0.0~~	BOOL			Salida rápida
Application.GVL.SALIDAS[1]		Q1	~~%QX0.1~~	BOOL			Salida rápida
Application.GVL.SALIDAS[2]		Q2	~~%QX0.2~~	BOOL			Salida rápida
Application.GVL.SALIDAS[3]		Q3	~~%QX0.3~~	BOOL			Salida rápida
Application.GVL.SALIDAS[4]		Q4	~~%QX0.4~~	BOOL			
Application.GVL.SALIDAS[5]		Q5	~~%QX0.5~~	BOOL			
Application.GVL.SALIDAS[6]		Q6	~~%QX0.6~~	BOOL			

Figura 4.2-13 *Asignación de entradas y salidas del PLC*

Esto mismo se hace para la asignación de E/S o de los PDO (Process Data Object) de los variadores en CANopen.

4.3 Programación del proyecto: VijeoDesigner

Gran parte de la dificultad de este proyecto residía en la realización de un interfaz gráfico que fuera asequible para cualquier operario que tenga que poner en marcha, cambiar alguno de los parámetros, o simplemente visualizar el funcionamiento de estas fuentes.

En este apartado se va a ver una explicación de las herramientas del software Vijeo Designer, que más se han utilizado para la realización de este proyecto, y un resumen del resultado de la programación en sí, haciendo un recorrido por las pantallas más relevantes. En el anexo 8.1

Manual para el usuario, que se ha desarrollado para el cliente, se pueden ver con más detalle todas las pantallas.

Vijeo Designer, es la aplicación que ha permitido crear los paneles de operador. Este programa proporciona todas las herramientas necesarias para el diseño de un proyecto HMI, desde la adquisición de datos hasta la creación y visualización de sinopsis animadas.

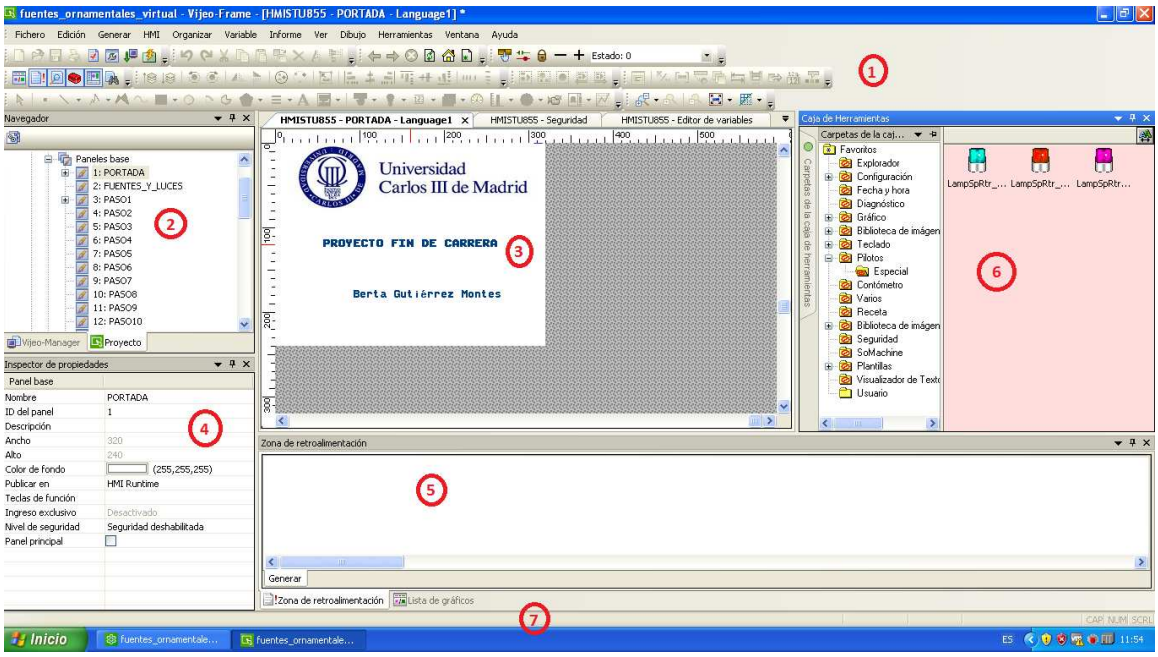


Figura 4.3-1 *Ventana principal Vijeo Designer V6.0*

Desde la ventana principal de Vijeo Designer, se puede acceder a las herramientas principales del programa. Cada ventana proporciona información relacionada con un objeto en particular o con el proyecto.

1. Menú de herramientas (Toolbar): para acceder a los menus y herramientas necesarios para desarrollar la aplicación.

2. Navegador (Navigator): utilizado para crear aplicaciones. La información relativa a cada proyecto se ordena de forma jerárquica en un explorador de documentos. Desde aquí se accede a todos los elementos de la aplicación (paneles, alarmas, emergentes…).

3. El editor gráfico (Graphic editor): espacio dónde se crea la aplicación.

4. El inspector de propiedades (Property inspector): zona dónde se configuran las propiedades de cualquier objeto. Muestra los parámetros del objeto seleccionado. Cuando se selecciona más de un objeto, sólo se muestran aquellos parámetros que son comunes a todos los objetos.

5. La zona de retroalimentación (Feedback zone): Muestra la progresión y los resultados de la comprobación de errores, de la compilación y de la carga.

Lista de gráficos (Graphics list): Ofrece una lista con todos los objetos que figuran en la sinopsis e indica:

- el orden de creación
- el nombre
- la posición
- las animaciones
- otras variables asociadas

6. La caja de herramientas (Toolchest): dónde encontrar los objetos gráficos (cronómetros, gráfico de barras, etc.) que suministra el fabricante o los que el usuario ha creado con anterioridad.

7. Barra de estado (Status bar): informa del estado del software (zoom, posición de las ventanas…)

El elemento fundamental de trabajo, dónde se muestra toda la información y dónde se recogen todos los parámetros necesarios, son los paneles gráficos.

En el nodo de paneles gráficos del Navegador, se encuentran los tres tipos de paneles existentes:

1. Paneles principales, dónde se dibujan objetos o se utilizan los de la caja de herramientas para crear un panel con objetos comunes que puedan utilizarse para todos o algunos paneles base del proyecto.

Figura 4.3-2 *Paneles Principales*

En el proyecto se han utilizado dos paneles principales. Uno simplemente con el símbolo del logout. Para poder acceder a determinados paneles de configuración será necesario identificarse con usuario y contraseña. Cuando el usuario termine podrá pulsar este icono para cerrar su sesión, también se ha configurado para que pasado un intervalo de tiempo, sin hacer uso de la pantalla, se cierre sesión automáticamente.

El otro panel principal, se utilizará en la configuración de los juegos de luces y chorros. Cómo se verá más adelante, se ofrece la opción de crear hasta diez pasos secuenciales diferentes de encendido y apagado de las distintas salidas. Para cada paso se ha creado un panel, pero en todos los paneles había botones e imágenes comunes, por ello se creó el panel principal de la figura anterior.

2. Paneles de ventana emergente, éstos se abrirán sobre el panel actual usando un interruptor, animación de contacto o un script.

Los paneles emergentes, se agrupan según la forma y el tamaño que se les dé. Para este proyecto se han utilizado estos paneles con distintos fines. En la figura siguiente se pueden ver algunos ejemplos utilizados de manera informativa, para asegurarse de que el usuario toma la decisión correcta, o simplemente para poner en marcha algún elemento de la fuente.

Figura 4.3-3 *Paneles emergentes*

3. Por último están los paneles base. Estos paneles se convertirán en la pantalla que aparece en las máquinas de destino.

Para este proyecto se han creado 31 paneles base para la configuración y visualización de la fuente.

4.3.1 **Configuración.**

Se distinguen dos tipos de configuraciones, la CONFIGURACIÓN BÁSICA, a la que todos los operarios podrán tener acceso, desde dónde se pueden ajustar determinados parámetros que no conllevan una parada de la fuente (ajuste de la hora, horarios de encendido de la depuradora…), o acceder a las visualizaciones. Y la CONFIGURACIÓN AVANZADA a la que sólo se puede acceder introduciendo un nombre de usuario y contraseña, restringiendo de este modo el acceso a determinados paneles de ajustes.

En el menú principal, o de CONFIGURACIÓN BÁSICA, además de poder acceder a las funciones de ANEMÓMETRO, HORARIOS, DEPURADORA o a las visualizaciones tanto de la fuente, como del nivel del vaso, se puede acceder al menú de ajustes avanzados.

Figura 4.3-4 *Panel Configuración Básica*

Para ello será necesario hacer un log in, introduciendo el nombre de usuario y contraseña autorizado. En ese momento se tendrá acceso a toda la definición de la fuente, necesario para su puesta en marcha.

Figura 4.3-5 *Acceso configuración avanzada*

La CONFIGURACIÓN AVANZADA o configuración inicial, se podrá hacer siguiendo el PASO A PASO o moviéndose libremente por cada panel.

Figura 4.3-6 *Menús configuración avanzada*

Configuración paso a paso:

El pulsador PASO A PASO, guiará al usuario por una serie de pantallas con las que se obtiene una configuración completa.

Para pasar de una pantalla a otra, en la parte inferior derecha de cada submenú se encuentran unas flechas que conducen al paso inmediatamente anterior o posterior. Estas flechas del PASO A PASO, sólo estarán visibles y por lo tanto accesibles en el caso de que se opte por la configuración PASO A PASO.

Las pantallas de la configuración PASO A PASO son las siguientes:

Configuración de variadores:

En esta pantalla se define el número de variadores y los parámetros de éstos.

- Número de variadores: limitado entre 0 y 8. Pulsando en RESET VVD se resetea el bus de comunicaciones, estableciendo de este modo la comunicación entre los variadores y la pantalla.

- Parámetros variador: frecuencias máxima, media y mínima de funcionamiento para cada variador.

- Copia variadores: en el supuesto de que todos los variadores tengan las mismas frecuencias máxima, media y mínima, pulsando este botón se transfieren los parámetros del primer variador a todos los demás.

Figura 4.3-7 *Panel configuración variadores*

Configuración del anemómetro:

En esta pantalla se define la velocidad del viento a la que los variadores disminuirán la frecuencia.

Los valores de velocidad mínima, media y máxima serán introducidos en m/s. Éstos valores están limitados al rango de valores que permite el anemómetro seleccionado (MTN580692) y que está conectado a la entrada digital del autómata (I1), de 0 a 40m/s.

Figura 4.3-8 *Panel configuración anemómetro*

El significado de estos tres valores es el siguiente:

Cuando se registre una velocidad inferior a la mínima, los variadores funcionaran a la frecuencia que configurada cómo máxima en el panel anterior, o a la que se indique en cada juego cómo se verá más adelante. Esto es, a la frecuencia normal de funcionamiento.

Una vez superado este valor, cuando la velocidad del viento se encuentre entre VELOCIDAD MÍNIMA y VELOCIDAD MEDIA, los variadores forzaran una reducción en la frecuencia hasta su FRECUENCIA MEDIA.

Del mismo modo, entre la VELOCIDAD MEDIA y VELOCIDAD MÁXIMA, se bajará a la FRECUENCIA MÍNIMA. Y en caso de que se supere la VELOCIDAD MÁXIMA, que en el indicador de VIENTO (parte inferior del panel) aparece con un marcador rojo (el amarillo y el naranja representan la mínima y media respectivamente), los variadores se detendrán.

Cuando la frecuencia se reduce, para volver a aumentar, los variadores habrán de esperar al menos durante el tiempo que se haya definido cómo mínimo retardo (RETARDO ENCENDIDO POR ELEVADO VIENTO). Una vez superado este tiempo y cuando vuelva a disminuir la velocidad del viento por debajo de la que esté condicionando, los variadores volverán a la velocidad que corresponda.

En todo momento se visualizará el valor de la velocidad del viento en el indicador de la parte inferior.

Configuración de las coordenadas GPS:

En esta pantalla únicamente se definen las coordenadas GPS dónde estará ubicada la fuente.

Para introducir los datos simplemente pulsando en el visualizador, aparecerá un teclado numérico.

Es importante introducir correctamente estos datos para el cálculo de las horas orto y ocaso, que se verá en el siguiente submenú.

Figura 4.3-9 *Panel coordenadas GPS*

Configuración de la depuradora:

La depuradora se activará todos los días, siempre y cuando el vaso no esté vacío. Desde este panel, se define a qué hora y durante cuánto tiempo estará activa.

Figura 4.3-10 *Panel configuración depuradora*

Cuando la depuradora esté activa se pueden ver las hélices de la pantalla moverse.

Configuración de los horarios:

Esta pantalla sirve para visualizar la fecha y hora del PLC y hacer los ajustes correspondientes.

Desde el submenú de horarios, se tiene acceso a tres acciones diferentes.

- CAMBIAR FECHA Y HORA
- HORARIO FUENTES
- HORARIO LUCES

Figura 4.3-11 *Panel configuración horarios*

Pulsando el botón central (CAMBIAR FECHA Y HORA), se accede al panel de AJUSTE DE HORA Y FECHA PLC.

En esta pantalla aparecen dos visores de fecha y hora. El de la parte inferior de la pantalla muestra la fecha y hora actual del PLC, y en el visor superior se introducirían los cambios.

Figura 4.3-12 *Panel ajuste fecha y hora*

Pulsando OK, se regresa al submenú HORARIOS.

Para configurar el horario de encendido y apagado, tanto de las luces cómo de las fuentes, se pulsa en los botones correspondientes, situados en la parte inferior de la pantalla HORARIOS.

Figura 4.3-13 *Panel configuración horarios fuentes*

Tanto en las fuentes como en las luces se podrán definir horarios distintos para los fines de semana (sábado y domingo) y para el resto de los días (LMXJV).

Pulsando sobre los visualizadores aparecerá un teclado numérico mediante el cual introducir los valores deseados. Desde qué momento hasta qué momento se quiere que estén operativos fuentes y luces.

Una vez introducidos los cambios, pulsando OK, se regresa al submenú HORARIOS. Los valores que el autómata trae configurados por defecto, son para que tanto luces cómo fuentes estén activos las 24 horas del día.

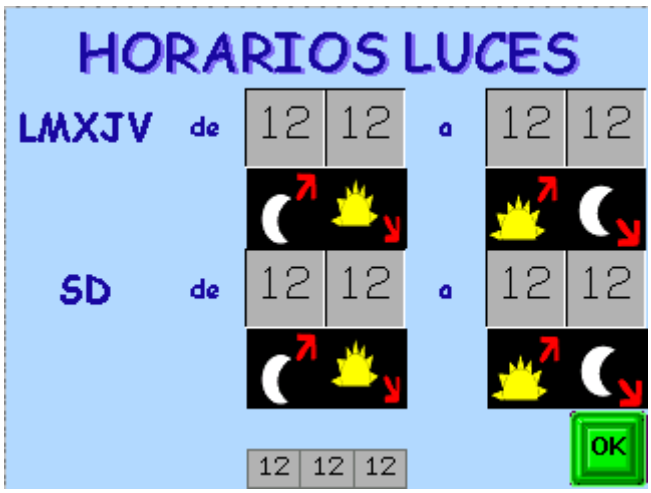


Figura 4.3-14 *Panel configuración horarios luces*

La única diferencia entre la configuración de las fuentes y la de las luces reside en los botones adicionales que tiene la pantalla luces. Estos pulsadores escribirán directamente, sobre la hora de encendido, el momento en el que tiene lugar la puesta de sol (ocaso), y sobre la hora de apagado, la hora del amanecer (orto). De este modo se consigue que las luces sólo se enciendan cuando sea de noche. Si está activo, no se puede cambiar la hora manualmente, y la hora de encendido y apagado, irá cambiando con el paso de los días. Si en caso contrario estuviera inactivo, pulsando el pulsador 

(que aparecerá cuando la opción de orto o de ocaso esté activa), se podrá sobrescribir la hora que se desee.

La condición de que sólo se activen las luces cuando sea de noche, también se puede poner por programa directamente.

Configuración de las salidas:

Siguiendo por el PASO A PASO, se llegará al último punto de la configuración. La asignación de las salidas digitales.

Figura 4.3-15 *Panel configuración fuentes*

En el submenú FUENTES Y LUCES, aparecen las 15 salidas digitales de las que dispone la fuente. En este punto habrá que definir cuáles de éstas corresponden a las luces y cuáles a las bombas de agua.

En los botones está indicado el número de la salida física a la que el usuario se está refiriendo. Cada toque que se aplica a los pulsadores cambiará la salida según la secuencia cíclica “desconectada-bomba-luz”.

Figura 4.3-16 *Pulsador opciones salidas*

Son diferentes la primera y segunda salida en la que además de bomba o luz se pueden definir, la primera cómo depuradora y la segunda cómo extractor en el caso de que hubiera.

Una vez finalizada la asignación de las salidas, para ponerla en marcha, y definir los juegos de encendido y apagado que se desee que haga la fuente, se pulsará el botón CONFIG FTE. En este momento se abrirá un panel emergente para confirmar que se quiere parar la fuente, ya que será necesario hacerlo para cambiar las salidas (este botón borrará los juegos que hubiera configurados).

Por otro lado el pulsador CAMBIO JUEGOS, da la opción de modificar la secuencia de juegos sin borrar la anterior, escribiendo sobre esta las modificaciones deseadas.

Submenús configuración:

Cómo ya se dijo al principio de este apartado, aunque sea recomendable, al menos la primera vez que se configure la fuente, seguir la configuración PASO A PASO, no es requisito indispensable hacerlo de este modo.

Se puede acceder a cada apartado de forma directa desde los dos menús de configuración:

- FREQ. VARIADORES (MENÚ CONFIGURACIÓN AVANZADA)
- ANEMÓMETRO (MENÚ CONFIGURACIÓN BÁSICA)
- HORARIOS (MENÚ CONFIGURACIÓN BÁSICA)
- FUENTES Y LUCES (MENÚ CONFIGURACIÓN AVANZADA)

Los botones de VISUALIZACIÓN, conducen a una visualización desde dónde puede supervisar y controlar el funcionamiento y estado de la fuente cómo se verá más adelante.

4.3.2 **Funcionamiento**

La fuente podrá funcionar de dos modos, manualmente o automáticamente haciendo juegos. Esto es, siguiendo una secuencia de encendidos, apagados o de cambios de frecuencia (esto último, sólo en el caso de los variadores). Directamente desde la pantalla se puede definir el número de pasos (hasta 10) que forman la secuencia cíclica, tiempo de cada paso y qué salidas se quiere que estén activas o no en cada paso.

En los paneles de definición de cada paso, sólo se verán las salidas que estén conectadas y se verá de qué tipo de salida se está hablando, según lo último que se haya definido en la pantalla de asignación de salidas.

Figura 4.3-17 *Panel configuración pasos*

En todo momento se puede saber qué paso se está configurando, gracias al visualizador PASO. Y debajo de éste se debe introducir el tiempo de cada paso en segundos.

Para definir qué salidas están activas y cuáles no, pulsando en el botón que indica el estado (ON/OFF), se podrá cambiar. De este modo quedarán definidos los juegos de las salidas digitales (luces y bombas), para modificar las frecuencias de los variadores, pulsar la tecla que indica JUEGOS VARIADORES.

Esta acción abrirá un panel emergente, en el cual se puede modificar la frecuencia de tantos variadores cómo se tengan conectados y se hayan configurado.

Para acceder a la frecuencia pulsando sobre el botón que se refiere al variador deseado y aparece el visualizador con la frecuencia que tenga en ese momento por defecto 50Hz. Y desde aquí se podrá modificar independientemente del resto de variadores. Habrá que pulsar sobre el visualizador e introducirla a través del teclado.

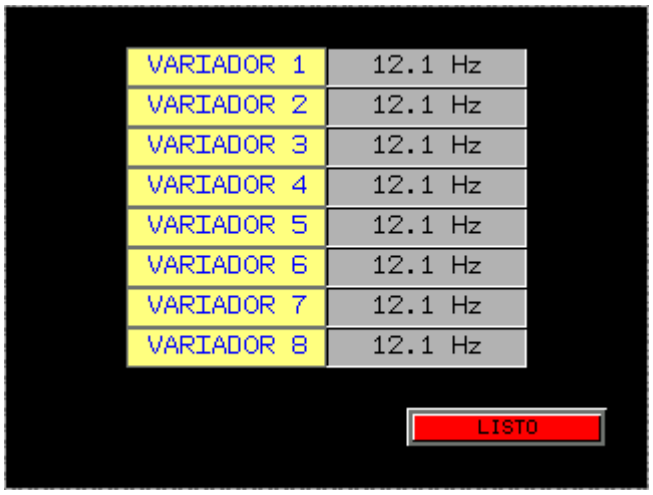


Figura 4.3-18 *Panel frecuencias variadores*

Una vez hechas todas las modificaciones deseadas, pulsando LISTO se volverá a la pantalla anterior.

En la parte inferior de la pantalla de definición de los pasos hay cuatro botones además del MENU.

Los botones de ANTERIOR y SIGUIENTE, permiten moverse por los distintos pasos. Desde el paso 1, cómo excepción, si se pulsa el botón ANTERIOR se dirigirá de nuevo al panel de configuración de las salidas.

Los otros dos botones son para poner en marcha la fuente. Existen dos métodos para poner en marcha la fuente:

Manual

Pulsando este botón, el encendido será controlado, además será sin juegos de encendido y apagado. Desde el panel de PUESTA EN MARCHA, por un lado se activan las salidas digitales y por otro los variadores.

Las salidas digitales se pueden activar o desactivar con el pulsador de cada una de ellas o se puede hacer de un modo más directo pulsando el botón verde de MARCHA. Si se hace de este segundo modo, se activarán todas las salidas que hayan sido configuradas, progresivamente, del siguiente modo: primero se activan todas las luces, y a continuación se irán activando uno a uno los contactores de las bombas, dejando entre ellos un tiempo también configurable por pantalla con el parámetro TIEMPO ENTRE CONTACTORES, siendo por defecto de 500ms.

Figura 4.3-19 *Panel puesta en marcha*

A continuación, pulsando el botón de VAR, se abre un nuevo panel emergente (sólo aparecerán los pulsadores de los que variadores hayan sido seleccionados):

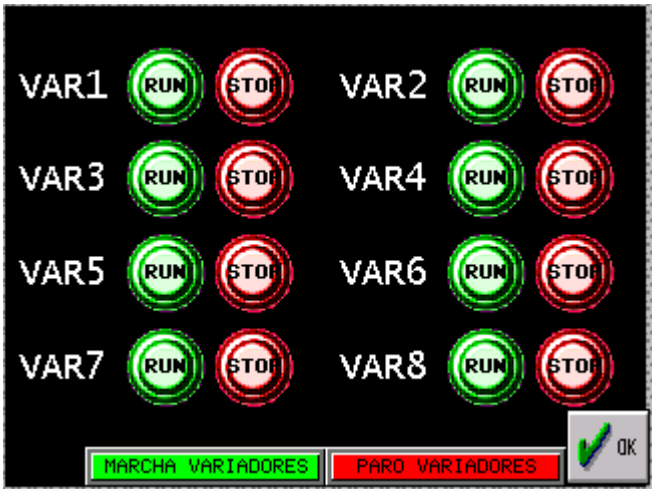


Figura 4.3-20 *Panel manualización variadores*

Desde este panel se encienden y apagan los variadores de uno en uno pulsando el botón correspondiente, a cada uno de ellos, o todos a la vez con el botón MARCHA o PARO VARIADORES.

Cuando estén todos funcionando pulsando OK este emergente se cerrará.

Inicio

Pulsando este botón, las salidas se encenderán automáticamente cuando les corresponda respetando la secuencia de juegos definida. Las fuentes y los variadores siempre lo harán de un modo progresivo, primero las bombas (salidas digitales, y a continuación los variadores).

En el momento en que se pulse el botón se activarán las salidas digitales y los variadores (con las frecuencias correspondientes) configuradas en el paso1, y empezará a correr la secuencia, siempre que los horarios de encendido lo permitan.

Automáticamente con la activación de los juegos la pantalla mostrará los paneles de visualización que se verán en la siguiente sección.

4.3.3 **Visualización**

En los paneles de visualización, se puede ver el estado actual de la fuente, bien de los variadores, de las luces y bombas, o del nivel de llenado del vaso de la fuente.

Visualización bombas y luces:

Existen dos modos de acceso, desde el menú principal, pulsando el botón VISUALIZACIÓN FUENTE, o de modo automático, cuando se pone la fuente en marcha, el programa, se redirige directamente hasta la visualización de las luces y bombas, y desde este se tiene acceso al de los variadores.

Para cambiar del panel de bombas y luces, al de variadores, se hará mediante la pulsación de la flecha azul, que indica a que visualización se dirige.

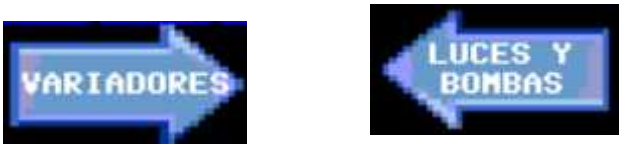


Figura 4.3-21 *Flechas cambio paneles*

En este panel se visualiza el estado de las salidas (luces/bombas) en cada momento, la hora actual y el paso de los juegos, en el que se encuentra.

Figura 4.3-22 *Panel visualización*

Si la salida está activa se verá de qué tipo de salida se trata, y si no en su lugar aparecerá el indicador de inactividad (OFF). Sólo aparecerán las salidas que hayan sido configuradas.

En la parte inferior del panel, se distinguen las siguientes funciones:

Además del botón que dirige al menú principal (MENU), aparecen los distintos pulsadores de parada.

El STOP, realiza una parada general, tanto de variadores como de luces y bombas, y redirecciona al menú principal.

Los cuatro pulsadores amarillos permiten parar o encender manualmente por separado las salidas digitales (Bombas y Luces) y los variadores.

Se puede ver en este panel, la flecha para acceder a la visualización de los variadores.

Visualización variadores:

Estos paneles no son sólo una visualización. Desde este panel además de los pulsadores de marcha y paro que tenía también la visualización anterior, desde aquí se puede también manipular el estado de cada variador individualmente.

Figura 4.3-23 *Visualización variadores*

El visualizador numérico, indica el número de variador al que se refiere el panel, y con las flechas inferiores se puede navegar por los variadores que se hayan configurado.

Este panel permite parar, poner en marcha o resetear cada variador por separado, con los pulsadores RUN, STOP, RESET, o pararlos y ponerlos en marcha todos a la vez cómo se vio con anterioridad, con los pulsadores STOP VARIADORES, RUN VARIADORES, de la parte inferior de la pantalla.

También permite como en el otro panel de visualización, parar y poner en marcha sólo las bombas y luces (STOP/RUN BOMBAS Y LUCES), o parar toda la fuente en su conjunto (STOP).

La información que ofrece esta visualización, de cada variador serán, la velocidad actual (FRECUENCIA) y la que se le está demandando (CONSIGNA) a través de las barras horizontales.

Además, en el otro gráfico de barras horizontal, se verá la velocidad del viento, que cómo ya se explicó condiciona la frecuencia a la que se quiere que funcionen los variadores. Los indicadores rojo, naranja y amarillo, que se pueden ver en este gráfico señalan la velocidad del viento que definida cómo máxima, media y mínima respectivamente, a partir de la cuál los variadores reducirán sus velocidades.

Del mismo modo que en el panel anterior, desde esta visualización se accede a la otra a través de la flecha azul, y al menú de inicio con el pulsador MENU.

Visualización nivel del vaso:

Este panel si que se trata de una simple visualización, en ella se puede supervisar en todo momento el nivel de agua del vaso de la fuente.

Se tienen tres sensores a lo largo del vaso, asociados a tres de las entradas digitales del autómata (I1, I2, I3), con lo que el vaso del panel se dividirá en cuatro franjas.

Cuando sólo quede un cuarto de éste, el indicador de nivel se pondrá naranja y parpadeará. Cuando baje de ese nivel, las fuentes se apagarán.

Figura 4.3-24 *Visualización nivel vaso*

Capítulo 5

Conclusiones y mejoras

El objetivo principal de este proyecto, era ofrecer una solución que contara cómo mínimo con todos los servicios que ofrecían las fuentes hasta ese momento.

Este objetivo se ha conseguido incluyendo además muchas mejoras que añaden valor a sus fuentes ornamentales.

Otro de los grandes retos de este proyecto era conseguir una interfaz gráfica que resultara intuitiva y accesible para cualquier operario. Se han tenido varias reuniones con el fabricante de fuentes y éste propuso mejoras que se han tenido en cuenta. En este punto el fabricante quedó satisfecho y con la ayuda del manual que se ha realizado, consideran que cualquier operario será capaz de poner la fuente en marcha. De todas formas seguramente podrían hacerse bastantes mejoras si se tuviera alguna reunión con algún operario directamente y se escucharan sus propuestas.

Se ha conseguido crear una arquitectura tanto hardware cómo software flexible a la vez que robusta.

Como protocolo de nivel superior para realizar la comunicación entre los distintos elementos conectados al bus se ha elegido CANopen, ya que está ampliamente extendido, ha sido adoptado cómo estándar internacional y para nuestra aplicación en concreto ha resultado mucho más estable que Modbus.

Otro de los objetivos cumplido sería conseguir una fuente más eficiente energéticamente, ya que con el uso de los variadores de velocidad cómo se vió en el subcapítulo de **Variación de velocidad,** el uso de estos aparatos garantiza un ahorro energético.

Este ahorro energético y económico también se consigue con algunas de las funciones desarrolladas:

- Gracias a la función orto-ocaso, que determina los momentos de luz de cada día del año, las luces nunca estarán encendidas mientras sea de día.
- La posibilidad de definir unos horarios de encendido para los días laborales y otro horario diferente para los fines de semana, permite recortar los horarios entre semana, podiendo disfrutar de un horario más extenso los fines de semana.
- La función anemómetro, prácticamente elimina la pérdida de agua por viento, siendo únicamente necesario reponer, el agua del circuito cerrado, que se pierda por evaporación.

Personalmente la realización de este proyecto ha sido una experiencia muy enriquecedora. Durante el desarrollo del proyecto me he sentido cada vez más cómoda con la programación. Mucho más ágil a la hora de encontrar fallos, o de hacer cualquier cambio o mejora.

Una de las mejoras que se podría implementar sería desarrollar un registro de alarmas, con posibilidad de visualizar estas por pantalla.

La otra mejora que se estuvo estudiando pero no se llevó a cabo porque encarecía bastante el presupuesto final, pero no se descarta la posibilidad de implementarla, como servicio adicional, sería añadir un módem GSM, para poder notificar y hacer alguna operación sencilla vía móvil.

Capítulo 6

Presupuesto

En este capítulo se detallan los costes derivados del diseño y construcción de una fuente ornamental.

Se incluyen tanto los gastos originados por la compra de los materiales hardware y software cómo los generados por las horas de ingeniería invertidas en el desarrollo del mismo.Todas las cantidades están expresadas en euros.

Los precios unitarios del material que aparecen en los siguientes apartados son orientativos y se pueden ver modificados en función del modo de compra, lugar de compra y número de unidades compradas.

6.1 Coste de material hardware

Referencia	Concepto	Cantidad	Precio unitario	Precio total
ATV312H037M2	ATV312 0,37KW 230V MONOFÁSICO	6	191,45	1148,7
ATV312H055M2	ATV312 0,55KW 230V MONOFÁSICO	1	211,45	211,45
VW3A31208	Carta ATV312 CanOpen daisy Chain	7	57,12	399,84
TM238LFDC24DT	CPU M238 24VDC 14E/10S CanOpen	1	464,1	464,1
HMISTU855	Terminal Táctil STU 5,7" Color	1	443,42	443,42
BMXXCAUSBH018	Cable USB industrial 1,8m	1	72,82	72,82
VW3A8306R10	Cable PLC-HMI MODBUS 2 RJ45 1m	1	6,55	6,55
TCSCCN4F3M05T	Cable CANOPEN 0,5m, SUBD9/RJ45	7	13,3	93,1
TCSCAR013M120	Adaptador final de linea CanOpen tipo RJ	1	11,03	11,03
TCSXCNAMUM3P	Cable programacion SoM 3m	1	78,54	78,54
ABL8REM24050	Fuente conmutada de 5A 24Vdc 120W	1	144,9	144,9
MTN580692	Sensor de viento blanco	1	91,92	91,92
	Total Hardware			3166,37

Tabla 6.1-1 *Coste Hardware*

6.2 Coste de material software

Referencia	Concepto	Cantidad	Precio unitario	Precio total
MSDCHNSFNV31 + MSDCHNLRUA	(CD Software) + Licencia. SoMachine y VijeoDesigner	1	392,7	392,7
	Total Software			392,7

Tabla 6.2-1 *Coste Software*

6.3 Coste de personal

La duración de este proyecto se ha estimado en 8 meses. En este tiempo se incluye el diseño, la implementación, y la redacción de la documentación del proyecto y de esta memoria.

Considerando el sueldo mensual bruto de 1500 euros.

Concepto	Sueldo mensual	Meses	Total
Ingeniero Industrial	1500	8	12000

Tabla 6.3-1 *Coste personal*

6.4 Presupuesto final

El importe total de desarrollo del proyecto asciende a una cantidad de:

Concepto	Precio total
Coste Material Hardware	3166,37
Coste Material Software	392,7
Coste Personal	12000
TOTAL	15559,07

Tabla 6.4-1 *Presupuesto total del proyecto*

QUINCE MIL QUINIENTOS CINCUENTA Y NUEVE EUROS CON SIETE CÉNTIMOS DE EURO.

Capítulo 7

Referencias

Capítulo 2: Estado del Arte de la Automatízación y Capítulo 3: Base Teórica

[1] Ilíada, libro XVIII, vv. 509-15

[2] Pausanias, Descripción de Grecia, libro II, capítulo IV.

[3] *Herón de Alejandría, The Pneumatics (Londres, 1851),* http://www.history.rochester.edu/steam/hero/index.html

[4] *Atlas Ilustrado de Los Robots de Leonardo, Colección "Bilbioteca Leonardo Da Vinci", Editorial Susaeta, 2010.*

[5] http://www.info-ab.uclm.es/labelec/solar/electronica, web de la Universidad de Castilla la Mancha, accedido 2012

[6] http://automatas.cps.unizar.es/Historia/Webs/automatas_en_la_historia.htm, web de la Universidad de Ingeniería Técnica Industrial de Zaragoza, accedido 2012

[7] *Historia de la Inteligencia Artificial,* http://matap.dmae.upm.es/LE/Divulgacion/IA/ISMAESWEB/historia/historia.htm, web de la Universidad Politécnica de Madrid, accedido 2012

[8] http://es.wikipedia.org/wiki/logica_cableada,

[9] http://es.wikipedia.org/wiki/automata_programable,

[10] *R. PIEDRAFITA MORENO: Ingeniería de la Automatización Industrial, Editorial Ra-Ma, 2003.*

[11] *E.MANDADO PEREZ, J. MARCOS ACEVEDO, C. FERNÁNDEZ SILVA, J.I. ARMESTO QUIROGA: Autómatas programables y sistemas de automatización, Editorial Marcombo, 2009.*

[12] *J. BALCELLS, J.L. ROMERAL: Autómatas programables, Editorial Marcombo,1997.*

[13] *Cursos Schneider Electric:* http://www.schneider-electric.com.ar/documents/recursos/myce/capitulo4_1907.pdf, Accedido 2012.

[14] *TCP/IP y el modelo OSI:* http://www.textoscientificos.com/redes/tcp-ip/comparacion-modelo-osi Accedido 2012.

Capítulo 4: Arquitectura y programación del proyecto

[15] http://www.schneider-electric.es

Fuentes ornamentales

[16] http://www.fuentesnovas.com/

[17] http://tramitarahora.ajuntamentdelx.es/CONTRATACIONYSERVICIOS/PROYECTOS/FONDOS%20MAP/Fuente%20ornamental%20Rotonda%20Avda%20de%20Santa%20Pola/Fuente%20ornamental.pdf

[18] http://www.aloj.us.es/notas_tecnicas/Mto_Fuentes_Ornamentales.pdf

[19] http://www.satta.es/site_flash/

Capítulo 8

Anexos

8.1 Manual para el usuario

8.2 Secuencias de código

8.3 Catálogos

8.4 Proyecto SoMachine